RLOTTO Code Manual

2018-03-02 22:08:26.971194

Contents

1 Makefile

```
1   # compile instructions from: https://github.com/prozumr/RLOTTO2/issues/1
2   # gcc rlotto.c t_select.c t_add.c t_evaluate.c t_delete.c settings.c t_common.c -o rlotto.exe
3
4   CC       = gcc
5   CFLAGS   = -Wall
6   BIN      = rlotto.exe
7   RLOTTOTEX = rlotto-manual.tex
8   LATEX    = lualatex
9   LATEXOPT = --interaction=batchmode -shell-escape
10
11  HEADER   = rlotto.h  version.h
12  CFILES   = rlotto.c  settings.c  t_add.c  t_common.c  t_delete.c \
13             t_evaluate.c  t_select.c
14  PYTHON   = create_rlotto_tex_file.py
15  OBJ      = rlotto.o t_select.o t_add.o t_evaluate.o t_delete.o settings.o \
16             t_common.o
17
18  all: $(OBJ)
19      $(CC)  -DMYSTRING='"hello"' $(CFLAGS) -o $(BIN) $(OBJ)
20
21  .PHONY: manual
22  manual:
23      cd manual; \
24      ./create_rlotto_tex_file.py \
25          --header  "$(HEADER)"   \
26          --csource "$(CFILES)"   \
27          --python  "$(PYTHON)"   \
28          > $(RLOTTOTEX);                  \
29      $(LATEX) $(LATEXOPT) $(RLOTTOTEX); \
30      $(LATEX) $(LATEXOPT) $(RLOTTOTEX)
31
32
33  .PHONY: clean
34  clean:
35      rm -rf $(BIN) $(OBJ)
36      cd manual && rm -rf *.aux *.log *.toc *.out manual/_minted*
```

2 Header files

rlotto.h

```
1   /*rlotto.h | RLotto | gcc | v0.8.354.1715
2    * Console program for storing and evaluating lottery ticket results.
3    * ----------------------------------------------------------------------
4    *
5    * Objective:    Providing global variables and functions for RLotto
6    *
7    * Author:       Reinhard Rozumek
8    * Email:        reinhard@rozumek.de
9    * Created:      10/08/17
10   * Last mod:     02/05/18
11   *
12   * ----------------------------------------------------------------------
13   * This file is part of RLotto.
14   *
15   * c is free software; you can redistribute it and/or
16   * modify it under the terms of the GNU General Public License
17   * as published by the Free Software Foundation; either version 3
18   * of the License, or (at your option) any later version.
19   *
20   * RLotto is distributed in the hope that it will be useful,
21   * but WITHOUT ANY WARRANTY; without even the implied warranty of
22   * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
23   * GNU General Public License for more details.
24   *
25   * You should have received a copy of the GNU General Public License
26   * along with RLOTTO2. If not, see <http://www.gnu.org/licenses/>.
27   */
28
29
30
31   #ifndef RLOTTO_H
32   #define RLOTTO_H
33   #define N 80
34   #define NOLR 12                                          // N_umber O_f L_ottery R_ows -1 (First Row with
35
36     /* GLOBAL VARIABLES */
37   bool go_Exit;
38   struct date { short month, day, year; };                // not used !? => check
39
40   extern int ALN[6];                                      // Actual Lottery Numbers
41   extern int ASN;                                         // Actual Super Number
42   extern char cG77[8];                                    // Actual Game 77 Number
43   extern char cSU6[7];                                    // Actual Super 6 Number
44   extern char TicketFolder[13];                           // Folder to store results
45   extern char ResultFolder[13];                           // Folder to store results
46   extern FILE *pFile;                                     // Pointer to result file
47
48   /* GLOBAL DEFINITIONS */
49   #define MAX_T_PLAYER_SZ 64                  // Max characters for player name
50   #define MAX_LINE_LENGTH 80                  // Max characters for length of a ticket row
51   #define T_EXT ".tck"                        // File extension for lottery tickets
52
53
54
55
56     /* FUNCTION DECLARATIONS */
57
58       int getche(void);
59       int terminateProgram(void);
60       int welcome(void);
61       int selectTicket(void);
62       int addTicket(void);
```

3

```c
        int evaluateTicket(void);
        int deleteTicket(void);
        int configureSettings(void);
        int terminateProgram(void);
        int t_initialize(void);                              // Initialize ticket data structure


/* STRUCTURE TICKET */

struct ticket {

        char T_No[8];                                 // Ticket Number - 7 digits + NULL terminator
        char T_Player[MAX_T_PLAYER_SZ];                 // Player Name(s). Site limited by MAX_T_PLAYER_SZ
        char T_Start[11];                             // Ticket Start Date
        char T_Runtime[2];                            // Runtime for ticket
        char T_D_Day[2];                            // Drawing Day: s_saturday, w_ednesday, or b_oth
        char T_G77[2];                               // Game 77 yes or no
        char T_SU6[2];                               // Super 6 yes or no
        char T_GSP[2];                               // Glueckspirale yes or no
        int T_Max_Row;                              // Max Number of active rows
        int T_Row[12] [6];                           // Lottery numbers per row

} current;


/* MAPPING ATTRIBUTES*/

        char display_runtime[10];                     // range: 1,2,3,4,5,month,permananet
        char display_weekday[10];                     // range: Wed,Sat,Wed + Sat
        char display_G77[4];                        // range: yes,no
        char display_SU6[4];                        // range: yes,no
        char display_GSP[4];                        // range: yes,no



#ifdef __linux__
        extern int lgetche(void);
#   define mygetc lgetche
#else
#   define mygetc getche
#endif

#endif // TICKET_H_INCLUDED
```

version.h

```
1   /*version.h | RLotto | gcc | v0.8.354.1715
2   * Console program for storing and evaluating lottery ticket results.
3   * ----------------------------------------------------------------------------
4   *
5   * Objective:      maintain version information for source code
6   *
7   * Author:         Reinhard Rozumek
8   * Email:          reinhard@rozumek.de
9   * Created:        10/08/17
10  * Last mod:       02/11/18
11  *
12  * ----------------------------------------------------------------------------
13  * This file is part of RLotto.                                           */
14
15
16  #ifndef VERSION_H
17  #define VERSION_H
18
19      // TODO (camelo#3#01/03/18): Improve handling of version number
20
21      //Program Name
22      static const char THISPROG[] = "RLOTTO";
23
24      //Software Status
25      static const char STATUS[] = "v0.8 beta";
26      static const char STATUS_SHORT[] = "v0.8b";
27
28      //Standard Version Type
29      static const long MAJOR = 0;
30      static const long MINOR = 8;
31      static const long BUILD = 354;
32      static const long REVISION = 1715;
33
34      //Miscellaneous Version Types
35      static const long BUILDS_COUNT = 354;
36      #define RC_FILEVERSION 0,8,353,1715
37      #define RC_FILEVERSION_STRING "0, 8, 354, 1715\0"
38      static const char FULLVERSION_STRING[] = "0.8.354.1715";
39
40      //These values are to keep track of your versioning state, don't modify them.
41      static const long BUILD_HISTORY = 29;
42
43
44  #endif //VERSION_h
45
46
47
48  /*
49
50  ABSTRACT
51
52  Originally generated by a Code::Blocks plugin. RLOTTO continues to use version.h
53  for central maintenance of versioning. However and since RLOTTO is developed with
54  no real IDE but just an editor (VIM/GVIM/Notepad++), FAR and the GNU Compiler this
55  file is maintained manually.
56
57
58  VERSION VALUES
59
60  Major - Increments by 1 when the minor version reaches its maximum
61  Minor - Increments by 1 when the build number pass the barrier of build times, the value is reset to 0 when it reach its max
62  Build number (also equivalent to Release) - Increments by 1 every time that the revision number is incremented.
63  Revision - Increments randomly when the project has been modified and then compiled.
64
65  STATUS
```

```
66
67     Some fields to keep track of your software status with a list of predefined values for convenience.
68
69     Software Status – The typical example should be v1.0 Alpha
70     Abbreviation – Same as software status but like this: v1.0a
71
72     SCHEME
73
74     Minor maximum – The maximum number that the Minor value can reach, after this value is reached the Major is incremented by 1
75     Build Number maximum – When the value is reached, the next time the project is compiled is set to 0. Put a 0 for unlimited.
76     Revision maximum – Same as Build Number maximum. Put a 0 for unlimited
77     Revision random maximum – The revision increments by random numbers that you decide, if you put here 1, the revision obvious
78     Build times before incrementing Minor – After successful changes to code and compilation the build history will increment, an
79     */
```

3 Sourcecode

rlotto.c

```
/*rlotto.c | RLotto | gcc | v0.8.354.1715
 * Console program for storing and evaluating lottery ticket results.
 * -----------------------------------------------------------------------
 *
 * Objective:     Main Menu and core program loop
 *
 * Author:        Reinhard Rozumek
 * Email:         reinhard@rozumek.de
 * Created:     09/23/17
 * Last mod:    02/11/17
 *
 * -----------------------------------------------------------------------
 * This file is part of RLotto.
 *
 * RLotto is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 3
 * of the License, or (at your option) any later version.
 *
 * RLotto is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with RLotto. If not, see <http://www.gnu.org/licenses/>.
 */



/* Header Section ********************************************************/

#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <dirent.h>
#include "rlotto.h"
#include "version.h"




/* GLOBAL VARIABLES ******************************************************/

int ALN[6];                                  // Actual Lottery Numbers
int ASN = -1;                                // Actual Super Number
char cG77[8];                                // Actual Game 77 Number
char cSU6[7];                                // Actual Super 6 Number
char TicketFolder[13] = ".\\tickets\\";      // Folder to store results
char ResultFolder[13] = ".\\results\\";      // Folder to store results
FILE *pFile = NULL;                          // Pointer to result file

/* Global Functions ******************************************************/

#ifdef __linux__
int lgetche(void)
{
    system("stty raw");//setting the terminal in raw mode
    char ch=getchar();
    system("stty cooked");
```

```
63          return(ch);
64      }
65      #endif
66
67      /****************************************************************************
68       * MAIN
69       ****************************************************************************/
70
71      /* Function main just displays the main user menu and generates the main
72          program loop. Program can only be terminated by the option from the main
73          menu. All submenus are coded in other functions. */
74
75      int main() {
76
77          int iSelect;                                        // Selection from Main Menu
78
79          welcome();                                          // Welcome Message
80          t_initialize();                                     // Initialize ticket data structure
81
82          // TODO (camelo#3#01/03/18): Implement Console attribute (Title, Color etc.) ...
83
84          /* Main Loop starts here */
85
86          do {
87
88              printf("\nMain Menu Options\n");
89              printf("-----------------------------------\n");
90              printf("\t1 - Add Lottery Ticket\n");
91              printf("\t2 - Delete Lottery Ticket\n");
92              printf("\t3 - Evaluate Lottery Ticket\n");
93              printf("\t4 - RLotto settings\n");
94              printf("\t5 - Terminate Program\n");
95
96              do {
97
98                  printf("\nPlease select (1-5): ");
99                  iSelect =(mygetc());
100                 fflush(stdin);
101
102             } while(iSelect < 49 || iSelect > 53);
103
104             iSelect = iSelect - 48;
105
106             switch(iSelect) {
107
108                 case 1: addTicket(); break;
109                 case 2: deleteTicket(); break;
110                 case 3: selectTicket(); break;
111                 case 4: configureSettings(); break;
112                 case 5: terminateProgram();
113
114             }
115
116         } while(go_Exit == false);
117
118     }
119
120     /****************************************************************************
121      * WELCOME
122      ****************************************************************************/
123
124     /* Welcome message and check for default RLotto file to open. */
125
126     int welcome(void) {
127
128         printf("\n%s v%ld.%ld.%ld.%ld\n",THISPROG,MAJOR,MINOR,BUILD,REVISION);
```

```c
129        printf("Evaluating lottery results\n") ;
130
131        return 0;
132    }
133
134    /***************************************************************************
135     * INITIALIZE TICKET STRUCTURE
136     ***************************************************************************/
137
138    /* Initializing all members of the ticket data structure.*/
139
140    int t_initialize(void) {
141
142        // Initialize ticket structure
143        strcpy(current.T_No, "0000000");
144        strcpy(current.T_Player, "000000000000000000000000000000000000000000000000000000000000000000");
145        strcpy(current.T_Start, "00/00/0000");
146        strcpy(current.T_Runtime, "0");
147        strcpy(current.T_D_Day, "u");
148        strcpy(current.T_G77, "u");
149        strcpy(current.T_SU6, "u");
150        strcpy(current.T_GSP, "u");
151        current.T_Max_Row = 0;
152
153        for(int j = 0; j < 12; j++) {
154            for(int k = 0; k < 6; k++) {
155                current.T_Row[j][k] = 0;
156            }
157        }
158
159        strcpy(display_runtime, "000000000");
160        strcpy(display_weekday, "000000000");
161        strcpy(display_G77, "000");
162        strcpy(display_SU6, "000");
163        strcpy(display_GSP, "000");
164
165        return 0;
166
167    }
168
169
170    /***************************************************************************
171     * TERMINATE PROGRAM
172     ***************************************************************************/
173
174    /* terminateProgram displays a message for the program termination. Other
175        functions like closing any open files, reset variables and write a history
176         message to the ini file still needs to be defined. */
177
178    int terminateProgram(void) {
179
180        printf("\n");
181         printf("\n\nProgram terminated by user.\n");
182         go_Exit = true; // remove after Main Menu is completed
183         return 0;
184    }
```

settings.c

```c
/*settings.c | RLotto | gcc | v0.0.2.0
 * Console program for storing and evaluating lottery ticket results.
 * ---------------------------------------------------------------------------
 *
 * Objective:    Configure settings and handling of ini file
 *
 * Author:        Reinhard Rozumek
 * Email:         reinhard@rozumek.de
 * Created:      10/08/17
 * Last mod:     10/08/17
 *
 * ---------------------------------------------------------------------------
 * This file is part of RLotto.                                          */


 // HEADER SECTION

#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <dirent.h>
#include "rlotto.h"

/* CONFIGURE SETTINGS ------------------------------------------------------*/
/* Configure settings in INI file.
*/
int configureSettings(void) {
    printf("\n\nConfigure Settings for RLotto.\n");


    printf("\n\n--- UNDER CONSTRUCTION ---\n");

    return 0;
}
```

t_add.c

```c
/*t_add.c | RLotto | gcc | v0.8.354.1715
* Console program for storing and evaluating lottery ticket resultt.
* -----------------------------------------------------------------------------
*
* Objective:    Create and store new ticket as ASCII file to file system.
*
* Author:          Reinhard Rozumek
* Email:           reinhard@rozumek.de
* Created:      10/08/17
* Last mod:     11/18/17
*
* -----------------------------------------------------------------------------
* This file is part of RLotto.                                            */


// HEADER SECTION

#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <ctype.h>
#include <time.h>
#include "rlotto.h"

//GLOBAL VARIABLES



// FUNCTION DECLARATION

int t_initialize(void);
void get_ticket_No(void);
void get_Player_Name(void);
void get_Start_Date(void);
void get_T_Runtime(void);
void get_T_D_Day(void);
void get_T_G77(void);
void get_T_SU6(void);
void get_T_GSP(void);
void get_T_Rows(void);
void display_Ticket(void);
void write_Ticket(void);
void map_t_attributes(int choice);
bool isCorrectTicketNo(char *n);
bool isCorrectDateFormat(int m, int d, int y);
bool isLeapYear(const int iYear);


/****************************************************************************
* ADD TICKET - CALL INPUT SUBMENUS
****************************************************************************/

int addTicket(void) {

    // TODO (camelo#2#01/20/18): Implement condition to add only enabled data (e.g. Super6, Spiel 77 etc.)

    bool is_ok = false;
    bool first_input = true;
    char sConfirm[2], sWrite[2];

    // CONFIRM TO CREATE NEW TICKET
    do {

```

```c
66          if(first_input == true)
67              printf("\n\nAdd new lottery ticket? (y/n): ");
68          else
69              printf("Invalid input! Please correct: ");
70          scanf("%s", sConfirm);
71          fflush(stdin);
72
73          if(*sConfirm=='y' || *sConfirm=='n') {
74              is_ok = true;
75          }
76
77          first_input = false;
78
79      } while(is_ok == false);
80
81      // INPUT SUBMENUS - EACH INPUT QUERY ENCAPSULATED IN ITS OWN FUNCTION
82      if(*sConfirm=='y') {
83
84          t_initialize();
85          get_ticket_No();
86          get_Player_Name();
87          get_Start_Date();
88          get_T_Runtime();
89          get_T_D_Day();
90          get_T_G77();
91          get_T_SU6();
92          get_T_GSP();
93          get_T_Rows();
94          display_Ticket();
95
96          // CONFIRM TO WRITE TICKET
97          is_ok = false;
98          first_input = true;
99
100         do {
101
102             if(first_input == true)
103                 printf("\nWrite ticket to file sytem? (y/n): ");
104             else
105                 printf("Invalid input! Please correct: ");
106             scanf("%s", sWrite);
107             fflush(stdin);
108
109             if(*sWrite=='y' || *sWrite=='n') {
110                 is_ok = true;
111             }
112
113             first_input = false;
114
115         } while(is_ok == false);
116
117         // WRITE TICKET TO FILE SYSTEM
118         if(*sWrite=='y') {
119
120             write_Ticket();
121
122         } else {
123
124             printf("\nCreation of new ticket canceled.\n");
125         }
126
127     } else {
128
129         printf("\nCreation of new ticket canceled.\n");
130     }
131
```

```
132        return 0;
133
134   }
135
136   /****************************************************************************
137    * GET TICKET NUMBER
138    ****************************************************************************/
139
140   /*  Gets Ticket Number by user input from stdin. */
141
142
143   void get_ticket_No(void) {
144
145       bool is_ok = false;              // correctness of input format
146       bool first_input = true;          // indicates first attempt for input
147       char ticket_No[8];                  // Ticket No to return to calling function
148
149       do
150       {
151           if(first_input == true)
152               printf("Enter 7-digit Ticket Number: ");
153           else
154               printf("Invalid input! Please correct: ");
155           scanf("%8s", ticket_No);                          //Input limited to 7 characters
156           fflush(stdin);
157           is_ok = (isCorrectTicketNo(ticket_No));
158           first_input = false;
159
160       } while(is_ok == false);
161
162       /* Remove trailing newline, if there and add 0 terminator. */
163       if ((strlen(ticket_No)>0) && (ticket_No[strlen (ticket_No) - 1] == '\n'))
164           ticket_No[strlen (ticket_No) - 1] = '\0';
165
166       strcpy(current.T_No , ticket_No);                    // Assign ticket number to structure
167
168   }
169
170   /****************************************************************************
171    * GET PLAYER NAME(S)
172    ****************************************************************************/
173
174   /*  Gets Player Name by user input from stdin. */
175
176   void get_Player_Name(void) {
177
178       char cPlayers[MAX_T_PLAYER_SZ];
179
180       printf("Enter Player Name: ");
181       fgets (cPlayers, MAX_T_PLAYER_SZ, stdin);
182
183       /* Remove trailing newline, if there and add 0 terminator. */
184       if ((strlen(cPlayers)>0) && (cPlayers[strlen (cPlayers) - 1] == '\n'))
185           cPlayers[strlen (cPlayers) - 1] = '\0';
186
187       strcpy(current.T_Player, cPlayers);                   // Assign ticket number to structure
188
189   }
190
191   /****************************************************************************
192    * GET TICKET START DATE
193    ****************************************************************************/
194
195   /*  Gets Date of purchase by user input from stdin. */
196
197   void get_Start_Date(void) {
```

```
198
199        struct tm ts;                   // date type for t_icket s_tart date
200
201        int year, month, day;           // year, month, day as enterd by user
202        bool is_ok = false;             // correctness of date format
203        char sPlayDate[11];             // Actual Drawing Date
204        bool first_input = true;    // indicates first attempt for input
205
206        do
207        {
208            if(first_input == true)
209                printf("Enter playing date (mm/dd/yyyy): ");
210            else
211                printf("Invalid input! Please correct: ");
212            scanf("%d/%d/%d", &month, &day, &year);
213            fflush(stdin);
214            is_ok = (isCorrectDateFormat(month, day, year));
215            first_input = false;
216
217        } while(is_ok == false);
218
219        ts.tm_year = year - 1900;
220        ts.tm_mon  = month - 1;
221        ts.tm_mday = day;
222
223        ts.tm_hour = 0;
224        ts.tm_min  = 0;
225        ts.tm_sec  = 1;
226        ts.tm_isdst = -1;
227
228        if ( mktime(&ts) == -1 )
229          ts.tm_wday = 7;
230
231        // strftime(sPlayDate, 40, "%A, %d-%B-%Y", &ts); <----
232
233        strftime(sPlayDate, 11, "%d.%m.%Y", &ts);
234
235
236        strcpy(current.T_Start , sPlayDate);                        // Assign ticket Start Date to structure
237
238        is_ok = false;          // reset to false for next evaluation
239        first_input = true;     // reset to true for next evaluation
240
241    }
242
243    /***************************************************************************
244    * GET TICKET RUNTIME
245    ***************************************************************************/
246
247    /* Gets ticket runtime in weeks by user input from stdin.
248       Allowed ranged is 1-5 (weeks),'m' (month) or 'p' (permanent) */
249
250
251    void get_T_Runtime(void) {
252
253        char rt[2];                     // Ticket runtimet in weeks .
254        bool is_ok = false;             // correctness of date format
255        bool first_input = true;    // indicates first attempt for input
256
257        do
258        {
259            if(first_input == true)
260                printf("Enter ticket runtime (1-5,m,p): ");
261            else
262                printf("Invalid input! Please correct: ");
263            scanf("%s", rt);
```

```
264            fflush(stdin);
265            if(*rt=='1' || *rt=='2' || *rt=='3' || *rt=='4' || *rt=='5' || *rt=='m' || *rt=='p' ) {
266
267                is_ok = true;
268
269                /* Remove trailing newline, if there and add 0 terminator. */
270                if ((strlen(rt)>0) && (rt[strlen (rt) - 1] == '\n'))
271                    rt[strlen (rt) - 1] = '\0';
272
273                strcpy(current.T_Runtime, rt);                    // Assign ticket runtime to structure
274            }
275
276            first_input = false;
277
278        } while(is_ok == false);
279    }
280
281    /***************************************************************************
282     * GET TICKET DRAWING DATE OF THE WEEK
283     ***************************************************************************/
284
285    /* Gets ticket drawing day of the week by user input from stdin.
286       Allowed ranged is s (Saturday), w (Wednesday) or b (Both - Saturday and
287       Wednesday) */
288
289    void get_T_D_Day(void){
290
291        char T_D_Day[2];             // Ticket runtimet in weeks .
292        bool is_ok = false;              // correctness of date format
293        bool first_input = true;     // indicates first attempt for input
294
295        do
296        {
297            if(first_input == true)
298                printf("Enter ticket drawing day (s,w,b): ");
299            else
300                printf("Invalid input! Please correct: ");
301            scanf("%s", T_D_Day);
302            fflush(stdin);
303            if(*T_D_Day=='s' || *T_D_Day=='w' || *T_D_Day=='b') {
304
305                is_ok = true;
306
307                /* Remove trailing newline, if there and add 0 terminator. */
308                if ((strlen(T_D_Day)>0) && (T_D_Day[strlen (T_D_Day) - 1] == '\n'))
309                    T_D_Day[strlen (T_D_Day) - 1] = '\0';
310
311                strcpy(current.T_D_Day, T_D_Day);                    // Assign ticket drawing day to structure
312            }
313
314            first_input = false;
315
316        } while(is_ok == false);
317    }
318
319    /***************************************************************************
320     * GET GAME 77
321     ***************************************************************************/
322
323    /* Gets info by user if ticket is enabled for "Game 77" (German "Spiel 77").
324       Allowed ranged is y (yes) or n (no) */
325
326    void get_T_G77(void){
327
328        char T_G77[2];                  // Indicates if Game 77 is active (y/n).
329        bool is_ok = false;             // Indicates correctness of date format
```

```
330        bool first_input = true;     // indicates first attempt for input
331
332        do
333        {
334            if(first_input == true)
335                printf("Enter if Game 77 is active (y,n): ");
336            else
337                printf("Invalid input! Please correct: ");
338            scanf("%s", T_G77);
339            fflush(stdin);
340            if(*T_G77=='y' || *T_G77=='n') {
341
342                is_ok = true;
343
344                /* Remove trailing newline, if there and add 0 terminator. */
345                if ((strlen(T_G77)>0) && (T_G77[strlen (T_G77) - 1] == '\n'))
346                    T_G77[strlen (T_G77) - 1] = '\0';
347
348                strcpy(current.T_G77, T_G77);                  // Assign Game 77 (yes or no) to structure
349            }
350
351            first_input = false;
352
353        } while(is_ok == false);
354    }
355
356    /***************************************************************************
357     * GET SUPER 6
358     ***************************************************************************/
359
360    /* Gets info by user if ticket is enabled for "Super 6".
361       Allowed ranged is y (yes) or n (no) */
362
363    void get_T_SU6(void){
364
365        char T_SU6[2];              // Indicates if Super 6 is active (y/n).
366        bool is_ok = false;          // Indicates correctness of user input
367        bool first_input = true;     // indicates first attempt for input
368
369        do
370        {
371            if(first_input == true)
372                printf("Enter if Super 6 is active (y,n): ");
373            else
374                printf("Invalid input! Please correct: ");
375            scanf("%s", T_SU6);
376            fflush(stdin);
377            if(*T_SU6=='y' || *T_SU6=='n') {
378
379                is_ok = true;
380
381                /* Remove trailing newline, if there and add 0 terminator. */
382                if ((strlen(T_SU6)>0) && (T_SU6[strlen (T_SU6) - 1] == '\n'))
383                    T_SU6[strlen (T_SU6) - 1] = '\0';
384
385                strcpy(current.T_SU6, T_SU6);                  // Assign Super 6 (yes or no) to structure
386            }
387
388            first_input = false;
389
390        } while(is_ok == false);
391    }
392
393    /***************************************************************************
394     * GET GLUECKSPIRALE
395     ***************************************************************************/
```

```c
396
397     /* Gets info by user if ticket is enabled for "Glueckspirale".
398         Allowed ranged is y (yes) or n (no) */
399
400     void get_T_GSP(void){
401
402         char T_GSP[2];                   // Indicates if Glueckspirale is active (y/n).
403         bool is_ok = false;              // Indicates correctness of user input
404         bool first_input = true;      // indicates first attempt for input
405
406         do
407         {
408             if(first_input == true)
409                 printf("Enter if Glueckspirale is active (y,n): ");
410             else
411                 printf("Invalid input! Please correct: ");
412             scanf("%s", T_GSP);
413             fflush(stdin);
414             if(*T_GSP=='y' || *T_GSP=='n') {
415
416                 is_ok = true;
417
418                 /* Remove trailing newline, if there and add 0 terminator. */
419                 if ((strlen(T_GSP)>0) && (T_GSP[strlen (T_GSP) - 1] == '\n'))
420                     T_GSP[strlen (T_GSP) - 1] = '\0';
421
422                 strcpy(current.T_GSP, T_GSP);                    // Assign Glueckspirale (yes or no) to structure
423             }
424
425             first_input = false;
426
427         } while(is_ok == false);
428
429     }
430
431     /****************************************************************************
432      * GET LOTTERY NUMBERS
433      ****************************************************************************/
434
435     /* Get lottery numbers per row. 12 rows defined. Range of number is 1-49.*/
436
437     void get_T_Rows(void){
438
439         int T_Max_Row;                    // Max number of rows used in this ticket
440         int i, j;                         // Loop counter
441         int checksum;                     // Checksum to validate numbers are in range of 1-49
442         int countInput;                   // Used as count for input arguments of scanf per iteration
443         int n[12][6];                     // Lottery number array
444         bool is_ok = false;               // Indicates correctness of user input
445         bool first_input = true;      // indicates first attempt for input
446
447         // Query number of active rows (T_Max_Row)
448
449         do
450         {
451             if(first_input == true)
452                 printf("Enter number of active rows (1-12): ");
453             else
454                 printf("Invalid input! Please correct: ");
455             scanf("%d", &T_Max_Row);
456             fflush(stdin);
457             if(T_Max_Row > 0 && T_Max_Row < 13) {
458
459                 is_ok = true;
460                 current.T_Max_Row = T_Max_Row;                   // Assign Glueckspirale (yes or no) to structure
461             }
```

```
462
463            first_input = false;
464
465        } while(is_ok == false);
466
467        // Query Lottery Numbers (T_Row[][])
468
469        is_ok = false;                                    // reset loop control variables
470        first_input = true;                               // reset loop control variables
471
472        do
473        {
474            // Initialzing variables required to reset for each loop;
475            checksum = 0;
476
477            for(i = 0; i < 12; i++){
478                for(j = 0; j < 6; j++){
479                    n[j][j]=0;
480                }
481            }
482
483            if(first_input == true)
484                printf("Enter Lottery numbers per row separated by commas.\n");
485            else {
486                printf("Invalid input! Please correct!\n\n");
487            }
488
489
490            for(i = 0; i <= T_Max_Row - 1; i++) {
491
492                countInput = 0;
493
494                printf("Row %d: ", i + 1);
495                countInput = scanf("%d,%d,%d,%d,%d,%d", &n[i][0],&n[i][1],&n[i][2],&n[i][3],&n[i][4],&n[i][5]);
496                fflush(stdin);
497
498                if(countInput == 6){
499
500                    // Checks range of each mumber in the row
501                    for(j = 0; j < 6; j++){
502
503                        if(n[i][j] > 0 && n[i][j] < 50){
504
505                            checksum++;
506                            current.T_Row[i][j] = n[i][j];
507                        }
508                    }
509
510                }else {
511                    printf("\nInvalid number of arguments in input\n");
512                    checksum = 0;
513                    t_initialize();                               //re-initialie to ensure structure is clean
514                    break;
515                }
516
517                if(countInput == 6 && checksum != (i+1) * 6){
518                    printf("\nInvalid value - All lottery numbers have to be in range from 1 to 49 \n");
519                    checksum = 0;
520                    t_initialize();                               //re-initialize to ensure structure is clean
521                    break;
522                }
523            }
524
525            if(checksum == 6 * T_Max_Row){
526                is_ok = true;
527            }
```

```c
528
529          first_input = false;
530
531      } while(is_ok == false);
532  }
533
534
535  /**************************************************************************
536   * WRITE INPUT TO TICKET FILE
537   **************************************************************************/
538
539
540
541  void write_Ticket(void){
542
543      // construct filename
544      char t_full_path[45];
545      strcpy(t_full_path, TicketFolder);
546      strcat(t_full_path, current.T_No);
547      strcat(t_full_path, T_EXT);
548
549      printf("\nFull path: %s\n", t_full_path);
550
551
552      // Open file and write
553      FILE *fp;
554      fp = fopen(t_full_path, "w");
555
556      if(fp == NULL) {
557
558          printf("\nTicket folder missing. Try to create now...\n");
559              system("mkdir tickets");
560              pFile = fopen(t_full_path, "w");
561              if(pFile == NULL) {
562                  printf("Error opening %s for writing. Program terminated.", TicketFolder);
563                  abort();
564              } else {
565                  printf("Folder \"%s\" has been created.\n", TicketFolder);
566              }
567
568
569      } else {
570
571          fprintf(fp, "Ticket No:%s\n", current.T_No);
572          fprintf(fp, "Player:%s\n", current.T_Player);
573          fprintf(fp, "Play Date:%s\n", current.T_Start);
574          fprintf(fp, "Runtime:%s\n", current.T_Runtime);
575          fprintf(fp, "Weekday:%s\n", current.T_D_Day);
576          fprintf(fp, "Game 77:%s\n", current.T_G77);
577          fprintf(fp, "Super 6:%s\n", current.T_SU6);
578          fprintf(fp, "Glueckspirale:%s\n", current.T_GSP);
579          fprintf(fp, "Active Rows:%d\n", current.T_Max_Row);
580
581          for(int j = 0; j < 12; j++) {
582
583              fprintf(fp, "Row %2d:%d,%d,%d,%d,%d,%d\n",j+1, current.T_Row[j][0],current.T_Row[j][1],current.T_Row[j][2],curr
584          }
585
586          fclose(fp);
587
588          printf("\nTicket %s written.\n", t_full_path);
589      }
590
591  }
```

t_common.c

```
1   /*t_common.c | RLotto | gcc | v0.8.354.1715
2   * Console program for storing and evaluating lottery ticket results.
3   * ----------------------------------------------------------------------------
4   *
5   * Objective:    Common functions used in various parts of this program.
6   *
7   * Author:        Reinhard Rozumek
8   * Email:         reinhard@rozumek.de
9   * Created:     11/17/17
10  * Last mod:    02/04/18
11  *
12  * ----------------------------------------------------------------------------
13  * This file is part of RLotto.                                        */
14
15
16  // HEADER SECTION
17
18  #include <stdio.h>
19  #include <stdbool.h>
20  #include <stdlib.h>
21  #include <string.h>
22  #include <dirent.h>
23  #include <ctype.h>
24  #include <time.h>
25  #include "rlotto.h"
26
27  /* GLOABL VARIABLES ********************************************************/
28
29  struct tm dd;                                    // date type used for d_rawing d_ate
30
31
32  // FUNCTION DECLARATION
33
34
35  bool isCorrectTicketNo(char *n);
36  bool isCorrectDateFormat(int m, int d, int y);
37  bool isLeapYear(const int iYear);
38  void map_t_attributes(int choice);
39  void display_Ticket(void);
40
41
42  /**************************************************************************
43   * IsCorrectTicketNo
44   **************************************************************************/
45  // Checks length and character (only '0' to '9' are allowed)
46
47   bool isCorrectTicketNo(char *n) {
48
49       bool result = false;
50       int x;
51       int y = 0;
52
53       // check for characters - only 0-9 are allowed
54       for(int i = 0; i < 7; i++) {
55           x = n[i] - 0;
56           if(x > 47 && x < 58)
57               y++;
58           else
59               break;
60       }
61
62       if(y == 7)
63           result = true;
64       return result;
65   }
```

```
66
67    /***************************************************************************
68     * IsCorrectDateFormat
69     ***************************************************************************/
70
71    bool isCorrectDateFormat(int m, int d, int y) {
72
73        int ii = 0;          // incremented by 1 for each positive validation
74        int febdays;     // used for number of days depending on year
75
76        /* Checking month '''''''''''''''''''''''''''''''''''''''''''''''''' */
77        if(m > 0 && m < 13)
78            ii++;
79        else
80            printf("\nInvalid value for month!\n");
81
82        /* Checking day depending on month '''''''''''''''''''''''''''''''''' */
83
84        // Jan, Mar, May, Jul, Aug, Oct, Dec
85        if(m==1|| m==3 || m==5 || m==7 || m==8 || m==10 || m==12) {
86            if(d > 0 && d < 32)
87                ii++;
88            else
89                printf("\nInvalid value for day!\n");
90        }
91
92        // Feb (validating leap years)
93
94        if((isLeapYear(y)) == true)
95            febdays = 29;
96        else
97            febdays = 28;
98
99        if(m==2) {
100           if(d > 0 && d < (febdays + 1))
101               ii++;
102           else
103               printf("\nInvalid value for day!\n");
104       }
105
106       // Apr, Jun, Sep, Nov
107       if(m==4|| m==6 || m==9 || m==11) {
108           if(d > 0 && d < 31)
109               ii++;
110           else
111               printf("\nInvalid value for day!\n");
112       }
113
114       /* Checking year '''''''''''''''''''''''''''''''''''''''''''''''''' */
115       if(y > 999 && y < 9999)                            // to ensure 4 digits
116           ii++;
117       else
118           printf("\nInvalid value for year!\n");
119
120       /* Evaluating complete result '''''''''''''''''''''''''''''''''''''' */
121
122       if(ii == 3)
123           return true;
124       else
125           return false;
126
127   }
128
129   /***************************************************************************
130    * IsLeapYear
131    ***************************************************************************/
```

```
132
133   bool isLeapYear(const int iYear)
134   {
135       // Each year that could be devided by 4 without rest is a leap year.
136       // Exception: year could be devided by 100 without rest but not by 400.
137
138       if ((iYear % 400) == 0)
139           return true;
140       else if ((iYear % 100) == 0)
141           return false;
142       else if ((iYear % 4) == 0)
143           return true;
144
145       return false;
146   }
147
148   /***************************************************************************
149    * MAP TICKET ATTRIBUTES
150    ***************************************************************************/
151
152   /* Map some structure data first to some more human readable values for display on stdout.
153      This is required since the input data for some ticket attributes differs from the output
154      format. E.g. y => yes
155
156      1: maps      current.T_Runtime    => display_runtime
157      2: maps      current.T_D_Day        => display_weekday
158      3: maps      current.T_G77        => display_G77
159      4: maps      current.T_SU6        => display_SU6
160      5: maps      current.T_GSP        => display_GSP
161
162   */
163
164   void map_t_attributes(int choice) {
165
166
167       // range of allowed values
168       char runtime_o1[2] = "1", runtime_o2[2] = "2", runtime_o3[2] = "3",
169       runtime_o4[2] = "4", runtime_o5[2] = "5", runtime_o6[2] = "m", runtime_o7[2] = "p";
170       char weekday_o1 [2] = "s", weekday_o2 [2] = "w", weekday_o3 [2] = "b";
171       char T_G77_o1[2] = "y", T_G77_o2[2] = "n";
172       char T_SU6_o1[2] = "y", T_SU6_o2[2] = "n";
173       char T_GSP_o1[2] = "y", T_GSP_o2[2] = "n";
174
175        switch(choice) {
176
177            // Runtime
178            case 1:
179
180                if(strcmp(current.T_Runtime, runtime_o1) == 0) strcpy(display_runtime, "1 week");
181                else if(strcmp(current.T_Runtime, runtime_o2) == 0) strcpy(display_runtime, "2 weeks");
182                else if(strcmp(current.T_Runtime, runtime_o3) == 0) strcpy(display_runtime, "3 weeks");
183                else if(strcmp(current.T_Runtime, runtime_o4) == 0) strcpy(display_runtime, "4 weeks");
184                else if(strcmp(current.T_Runtime, runtime_o5) == 0) strcpy(display_runtime, "5 weeks");
185                else if(strcmp(current.T_Runtime, runtime_o6) == 0) strcpy(display_runtime, "month");
186                else if(strcmp(current.T_Runtime, runtime_o7) == 0) strcpy(display_runtime, "permanent");
187                else strcpy(display_runtime, "");
188
189            break;
190
191            // Weekday
192            case 2:
193                if(strcmp(current.T_D_Day, weekday_o1) == 0) strcpy(display_weekday, "Sat");
194                else if(strcmp(current.T_D_Day, weekday_o2) == 0) strcpy(display_weekday, "Wed");
195                else if(strcmp(current.T_D_Day, weekday_o3) == 0) strcpy(display_weekday, "Wed + Sat");
196                else strcpy(display_weekday, "");
197            break;
```

```
198
199            // Game 77
200            case 3:
201
202                if(strcmp(current.T_G77, T_G77_o1) == 0) strcpy(display_G77, "yes");
203                else if(strcmp(current.T_G77, T_G77_o2) == 0) strcpy(display_G77, "no");
204                else strcpy(display_G77, "");
205
206            break;
207
208            // Super 6
209            case 4:
210
211                if(strcmp(current.T_SU6, T_SU6_o1) == 0) strcpy(display_SU6, "yes");
212                else if(strcmp(current.T_SU6, T_SU6_o2) == 0) strcpy(display_SU6, "no");
213                else strcpy(display_SU6, "");
214
215            break;
216
217            // Glueckspirale
218            case 5:
219
220                if(strcmp(current.T_GSP, T_GSP_o1) == 0) strcpy(display_GSP, "yes");
221                else if(strcmp(current.T_GSP, T_GSP_o2) == 0) strcpy(display_GSP, "no");
222                else strcpy(display_GSP, "");
223
224            break;
225
226            default: ;
227
228        }
229    }
230
231
232    /****************************************************************************
233     * Display Ticket
234     ****************************************************************************/
235    /* Displays ticket on terminal to confirm selection. This function can be used
236    for new tickets before writing ticket to file system or for existing tickets.
237    A loop for cofirming the selection is NOT part of the function itself. Note
238    that this function requires that the actual ticket data to display are already
239    expected to be in the current global ticket structure. */
240
241    void display_Ticket(void) {
242
243        int i;
244
245        /* Map some structure data first to some more human readable values for display on stdout */
246
247        map_t_attributes(1);     // Runtime
248        map_t_attributes(2);     // Weekday
249        map_t_attributes(3);     // Game 77
250        map_t_attributes(4);     // Super 6
251        map_t_attributes(5);     // Glueckspirale
252
253        // Display ticket --------------------------------------
254
255        printf("\n=================================================================================\n\n");
256        printf("  Ticket No: %-11sPlayer: %-26s Active Rows: %-10d\n", current.T_No, current.T_Player, current.T_Max_Row);
257        printf("  Date: %-16sTicket Runtime: %-19sWeekday: %-10s\n", current.T_Start, display_runtime, display_weekday);
258        printf("  Game 77: %-13sSuper 6: %-26sGlueckspirale: %-10s\n", display_G77, display_SU6, display_GSP);
259        printf("\n=================================================================================\n\n");
260
261        for(i = 0; i < 12; i++) {
262
263            printf("  Row %3d: %3d %3d %3d %3d %3d %3d\n",i+1, current.T_Row[i][0],current.T_Row[i][1],current.T_Row[i][2],curr
```

```
264         }
265
266         printf("\n=================================================================================\n\n");
267
268     }
269
270     /**************************************************************************
271     * IsCorrectLotteryRow
272     **************************************************************************
273     checks for valid range and duplicates                                */
274
275     bool isCorrectLotteryRow(int *LotteryNo) {
276
277         int i, j; int v = 0; bool bDuplicate = false;
278
279         // checks for duplicates
280         for(i = 0; i < 6; i++) {
281             for(j = 0; j < 6; j++) {
282                 if(i != j)
283                     if(LotteryNo[i] == LotteryNo[j])
284                         bDuplicate = true;
285             }
286         }
287
288
289         // checks for valid range
290         for(i = 0; i < 6; i++) {
291             if(LotteryNo[i] > 0 &&  LotteryNo[i] < 50)
292                 v++;
293         }
294
295         if((v == 6) && (bDuplicate == false))
296             return true;
297
298         return false;
299     }
300
301     /**************************************************************************
302     * Convert To Digit
303     **************************************************************************
304     converts single char in range of '1' to '9' to number.                */
305
306     /* returns int value if in range '0'..'9' else returns -1 if not a number */
307
308     int convertToDigit( char c )
309     {
310         if ( c < '0' || c > '9' ) return -1;
311         return c - '0';
312     }
313
314     /**************************************************************************
315     * Get Lottery Win Class
316     **************************************************************************
317     Function gets number of lottery matches per row and result from check of super
318     super number per ticket. The fuction returns the lottery win class.      */
319
320     char *getWinClass(int matches, bool super ) {
321
322         if(super == false) {
323
324             switch(matches) {
325
326                 case 0: return "no win"; break;
327                 case 1: return "no win"; break;
328                 case 2: return "no win"; break;
329                 case 3: return "class VIII"; break;
```

```
330                  case 4: return "class VI"; break;
331                  case 5: return "class IV"; break;
332                  case 6: return "class II"; break;
333                  default: return "no win";
334              }
335
336         } else {
337
338              switch(matches) {
339
340                  case 0: return "no win"; break;
341                  case 1: return "no win"; break;
342                  case 2: return "class IX"; break;
343                  case 3: return "class VII"; break;
344                  case 4: return "class V"; break;
345                  case 5: return "class III"; break;
346                  case 6: return "class I"; break;
347                  default: return "no win";
348              }
349         }
350  }
351
352
353    /****************************************************************************
354     * IsValidDrawingDate
355     ****************************************************************************
356     Check for ticket validity against drawing date. Gets Drawing Date and returns
357     true or false by checking Ticket Start Date, Ticket Runtime and Ticket day of
358     week – all stored in the global ticket structure. In order to accomplish this,
359     the functions first needs to calculate the ticket end date. */
360
361  bool isValidDrawingDate(int dd_month, int dd_day, int dd_year) {
362
363      bool result;                     // true if w_day_OK AND period_OK true. Otherwise false.
364      bool w_day_OK;                    // true if drawing day of the week matches with ticket. Otherwise false.
365      bool period_OK;                   // true if with ticket validity period. Otherwise false.
366      struct tm add;                   // date type used for a_ctual d_rawing d_ate
367      struct tm tsd;                   // date type used for t_icket s_tart d_ate
368      struct tm ted;                   // date type used for t_icket e_nd d_ate
369      int ts_day, ts_month, ts_year;   // representing year, month, day from ticket start date as integer values
370      char *str_day;                   // Token of Ticket start date for day
371      char *str_month;                 // Token of Ticket start date for month
372      char *str_year;                  // Token of Ticket start date for year
373      char str_runtime;                // runtime range: 1,2,3,4,5,m_onth, p_ermanent
374      /* char str_d_day;              drawing day range: s_saturday, w_ednesday or b_oth*/
375      double diff_seconds1;            // time difference in seconds
376      double diff_seconds2;            // time difference in seconds
377      int dw_day;                      // drawing weekday (0,1,2,3,4,5,6)
378      char tw_day;                     // ticket week day (s, w or b)
379
380
381      result = false;
382
383
384      // build actual drawing date structure -------------------------------------
385
386      add.tm_year = dd_year – 1900;
387      add.tm_mon  = dd_month – 1;
388      add.tm_mday = dd_day;
389
390      add.tm_hour = 0;
391      add.tm_min  = 0;
392      add.tm_sec  = 1;
393      add.tm_isdst = –1;        // Change for Summer Time !?
394
395      if (mktime(&add) == –1 )
```

```
396                add.tm_wday = 7;
397
398        // build ticket start date structure --------------------------------------
399
400        str_day   = strtok(current.T_Start, ".");
401        str_month = strtok(NULL, ".");
402        str_year  = strtok(NULL, ".");
403
404        ts_day   = atoi(str_day);
405        ts_month = atoi(str_month);
406        ts_year  = atoi(str_year);
407
408        tsd.tm_year = ts_year - 1900;
409        tsd.tm_mon  = ts_month - 1;
410        tsd.tm_mday = ts_day;
411
412        tsd.tm_hour = 0;
413        tsd.tm_min  = 0;
414        tsd.tm_sec  = 1;
415        tsd.tm_isdst = -1;        // Change for Summer Time !?
416
417        if (mktime(&tsd) == -1 )
418            tsd.tm_wday = 7;
419
420        // build ticket end date structure ---------------------------------------
421
422        str_runtime = current.T_Runtime[0];
423        //str_d_day = current.T_D_Day[0];
424
425        ted.tm_year = ts_year - 1900;
426        ted.tm_mon  = ts_month - 1;
427        ted.tm_mday = ts_day;
428
429        ted.tm_hour = 0;
430        ted.tm_min  = 0;
431        ted.tm_sec  = 1;
432        ted.tm_isdst = -1;        // Change for Summer Time !?
433
434        switch (str_runtime) {
435
436            case '1': ted.tm_mday +=7;  mktime(&ted); break;    // 1 week = 7 days
437            case '2': ted.tm_mday +=14; mktime(&ted); break;    // 2 weeks = 14 days
438            case '3': ted.tm_mday +=21; mktime(&ted); break;    // 3 weeks = 21 days
439            case '4': ted.tm_mday +=28; mktime(&ted); break;    // 4 weeks = 28 days
440            case '5': ted.tm_mday +=35; mktime(&ted); break;    // 5 weeks = 35 days
441            case 'm': ted.tm_mon +=1;   mktime(&ted); break;    // 1 month = same date next month
442            case 'p': ted.tm_year +=25; mktime(&ted); break;    // permanent = 25 years
443            default: break;
444        }
445
446
447        // Check validity --------------------------------------------------------
448        // Condition 1: weekday 0=Sun, 1=Mon, 2=Tue, 3=Wed, 4=Thu, 5=Fri, 6=Sat
449        // Condition 2: Ticket end date is greater or equals actual drawing date
450        // Condition 3: Actual drawing date is grater or equals ticket start date
451        // Ticket has been validated against drawing date once all conditions above are evaluated true
452
453        diff_seconds1 = difftime(mktime(&add), mktime(&tsd));        // difference between actual drawing date and ticket start d
454        diff_seconds2 = difftime(mktime(&ted), mktime(&add));        // difference between ticket end date and actual drawing dat
455
456        dw_day = add.tm_wday;                                        // day of the week for actual drawing date from console inpu
457        tw_day = current.T_D_Day[0];                                 // day of the week read from ticket (w, s or b)
458
459        // Check for correct weekday (condition 1)
460
461        w_day_OK = false;
```

```c
462        period_OK = false;
463
464
465        switch (tw_day) {
466
467            case 's':
468
469                if(dw_day == 6) w_day_OK = true;
470                else printf("\nDrawing date is not a Saturday. Please enter correct drawing date.\n");
471                break;
472
473            case 'w':
474
475                if(dw_day == 3) w_day_OK = true;
476                else printf("\nDrawing date is not a Wednesday. Please enter correct drawing date.\n");
477                break;
478
479            case 'b':
480
481                if(dw_day == 3 || dw_day == 6) w_day_OK = true;
482                else printf("\nDrawing date is neither Wednesday nor Saturday. Please enter correct drawing date.\n");
483                break;
484
485            default:
486
487                printf("\nDay of Week from ticket not available. \n");
488                break;
489        }
490
491
492
493        // Check for correct validity period (condition 2 & 3)
494
495        if((diff_seconds1 >= 0) && (diff_seconds2 >= 0)) period_OK = true;
496        else printf("\nActual drawing date not in rage of valid ticket period. Please enter correct drawing date.\n");
497
498        // Total result for check of validity
499
500        if(w_day_OK == true && period_OK == true) result = true;
501        else result = 0;
502
503        /* Debug
504        printf("DEBUG add: %s\n", asctime(&add));
505        printf("DEBUG tsd: %s\n", asctime(&tsd));
506        printf("DEBUG ted: %s\n", asctime(&ted));
507        printf("DEBUG diff_seconds1: %f\n", diff_seconds1);
508        printf("DEBUG diff_seconds2: %f\n", diff_seconds2);
509        printf("DEBUG dw_day: %d\n", dw_day);
510        printf("DEBUG tw_day: %c\n", tw_day); */
511
512        return result;
513
514 }
```

t_delete.c

```c
/*t_delete.c | RLotto | gcc | v0.8.354.1715
 * Console program for storing and evaluating lottery ticket results.
 * ----------------------------------------------------------------------------
 *
 * Objective:    Delete tickets from repository
 *
 * Author:        Reinhard Rozumek
 * Email:         reinhard@rozumek.de
 * Created:      10/08/17
 * Last mod:     10/08/17
 *
 * ----------------------------------------------------------------------------
 * This file is part of RLotto.                                            */


 // HEADER SECTION

#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <dirent.h>
#include "rlotto.h"


/* DELETE TICKET -----------------------------------------------------------*/
/* Delete stored ticket in repository.
*/
int deleteTicket(void) {
    printf("\n\nDelete existing lottery ticket.\n");

    return 0;
}
```

t_evaluate.c

```
1    /*t_evaluate.c | RLotto | gcc | v0.8.354.1715
2    * Console program for storing and evaluating lottery ticket resultt.
3    * ----------------------------------------------------------------------------
4    *
5    * Objective:    Evaluate selected ticket based on corresponding draw results
6    *
7    * Author:        Reinhard Rozumek
8    * Email:         reinhard@rozumek.de
9    * Created:       10/08/17
10   * Last mod:      02/11/18
11   *
12   * ----------------------------------------------------------------------------
13   * This file is part of RLotto.                                          */
14
15
16   // HEADER SECTION
17
18   #include <stdio.h>
19   #include <stdbool.h>
20   #include <stdlib.h>
21   #include <ctype.h>
22   #include <string.h>
23   #include <dirent.h>
24   #include <time.h>
25   #include "rlotto.h"
26   #include "version.h"
27
28
29   /* GLOABL VARIABLES **********************************************************/
30
31   struct tm dd;                                      // date type used for d_rawing d_ate
32
33   /* FUCTION DECLARATION ******************************************************/
34
35   int enterInput(void);                                      // enter drawing results
36   int checkLotto(void);                                      // evaluate lottery result
37   int checkGame77(void);                                     // evaluate Game 77 result
38   int checkSuper6(void);                                     // evaluate Super 6 result
39   int checkGSP(void);                                        // evaluate Glueckspirale result
40   int isCorrectDateFormat(int m, int d, int y);              // validating date format
41   bool isValidDrawingDate(int dd_month, int dd_day, int dd_year); // validating date range
42   bool isCorrectLotteryRow(int *LotteryNo);                  // checks valid range & duplicates
43   int convertToDigit( char c );                              // converts single char in range of '1' to '9' to number.
44   char *getWinClass(int matches, bool bonus_super );         // returns win class based on lottery matches
45
46   /*    ************************************************************************
47       EVALUATE TICKET
48       ************************************************************************/
49
50   /* Enter actuals drawings results and evaluate against selected ticket. This
51      function create the output/result file and calls all sub functions for ticket
52      evaluation. */
53
54
55       // FIXME: (camelo#1#01/20/18): Adjust for new rules - remove Bonus number adjust win classed by Superzahl
56       // TODO: (camelo#2#01/03/18): Implement condition to evaluate only enabled options (e.g. G77)
57
58   int evaluateTicket(void) {
59
60       int sConfirm;                              // yes or no to confirm user input
61       char sPath[45];                            // Full pathname for result file
62       char sPrefix[14] = "Lotto-Result-";        // File prefix
63       char sPostfix[5] = ".txt";                 // File extension
64       char sDrwDate[25];                         // Drawing Date formatted as part of filename
65
```

```
66        printf("\n\nEnter actual drawing result.\n");
67
68        // Call function to enter drawing results
69        enterInput();
70
71        // Final confirmation for starting evaluation
72        printf("\nEvaluate results now? [y/n]: ");
73
74        do {
75            sConfirm = tolower(getchar());
76        } while(sConfirm != 'y' && sConfirm != 'n');
77
78        if(sConfirm == 'y') {
79
80            // Create Filename
81            strftime(sDrwDate, N, "%Y-%m-%d", &dd); strcpy(sPath, ResultFolder); strcat(sPath, sPrefix);
82             strcat(sPath, current.T_No); strcat(sPath, "-"); strcat(sPath, sDrwDate); strcat(sPath, sPostfix);
83
84            // Open result file for output
85            pFile = fopen(sPath, "w");
86            if(pFile == NULL) {
87                printf("\nResult folder missing. Try to create now...\n");
88                system("mkdir results");
89                pFile = fopen(sPath, "w");
90                if(pFile == NULL) {
91                    printf("Error opening %s for writing. Program terminated.", sPath);
92                    abort();
93                } else {
94                    printf("Folder \"%s\" has been created.\n", ResultFolder);
95                }
96            }
97
98            // Output to result file first part (header information)
99            fprintf(pFile, "%s v%ld.%ld.%ld.%ld\n", THISPROG,MAJOR,MINOR,BUILD,REVISION);
100           fprintf(pFile, "Evaluating lottery results\n");
101           fprintf(pFile, "Lottery Ticket No: %s\n", current.T_No);
102
103           fprintf(pFile, "\nPlayers: %s\n", current.T_Player);
104
105           strftime(sDrwDate, N, "%A, %d-%b-%Y", &dd);
106
107           fprintf(pFile, "\nDrawing Date: %s\n", sDrwDate);
108           fprintf(pFile, "Lottery numbers: %i %i %i %i %i %i\n",ALN[0],ALN[1],ALN[2],ALN[3],ALN[4],ALN[5]);
109           fprintf(pFile, "Super number: %i\n", ASN);
110           fprintf(pFile, "Game 77: %s\n", cG77);
111           fprintf(pFile, "Super 6: %s\n", cSU6);
112
113           fprintf(pFile, "\nLottery Matches on %s\n\n", sDrwDate);
114
115           // Evaluate results
116           checkLotto(); checkGame77(); checkSuper6();
117           printf("\nWithout any warranty.\n");
118
119           // Final output for result file
120           fprintf(pFile, "\nWithout any warranty.\n");
121
122            // Close result file
123           fclose(pFile);
124           printf("\nResults written to %s.\n", sPath);
125
126        }
127
128        return 0;
129   }
130
131
```

```c
132    /***************************************************************************
133     * Enter Input
134     ***************************************************************************/
135
136    int enterInput() {
137
138        // Actual Play Date (drawing date) ''''''''''''''''''''''''''''''''''''''''''
139
140        int year, month, day;           // year, month, day as enterd by user
141        bool is_ok = false;            //
142        bool date_format_ok = false;   // correctness of drawing date format
143        bool date_range_ok = false;    // validity of drawing date related to ticket
144        char sPlayDate[40];            // Actual Drawing Date
145        int i;                         // Actual Lottery Number Index
146        bool first_input = true;       // indicates first attempt for input
147        char sDrwDate[N];              // Drawing date formated
148
149
150        while(date_format_ok == false || date_range_ok == false)
151        {
152
153            if(first_input == true)
154                printf("\nEnter drawing date (mm/dd/yyyy): ");
155            else
156                printf("Invalid input for drawing date! Please correct: ");
157            scanf("%d/%d/%d", &month, &day, &year);
158            fflush(stdin);
159
160            dd.tm_year = year - 1900;
161            dd.tm_mon  = month - 1;
162            dd.tm_mday = day;
163
164            dd.tm_hour = 0;
165            dd.tm_min  = 0;
166            dd.tm_sec  = 1;
167            dd.tm_isdst = -1;       // Change for Summer Time !?
168
169            if (mktime(&dd) == -1 )
170                dd.tm_wday = 7;
171
172            strftime(sPlayDate, 40, "%A, %d-%b-%Y", &dd);
173
174            date_format_ok = isCorrectDateFormat(month, day, year);
175            date_range_ok = isValidDrawingDate(month, day, year);
176
177
178
179            first_input = false;
180
181        }
182
183
184        first_input = true;    // reset to true for next evaluation
185
186        // Actual Lottery Numbers '''''''''''''''''''''''''''''''''''''''''''''''''''
187
188        do {
189            // Initialize ALN
190            for(i = 0; i < 6; ++i) {
191                ALN[i] = 0;
192            }
193
194
195            if(first_input == true)
196                printf("Enter actual lottory numbers seperated by commas: ");
197            else
```

```c
198                printf("Invalid input! /Lottery Numbers) Please correct: ");
199            scanf("%i,%i,%i,%i,%i,%i", &ALN[0], &ALN[1], &ALN[2], &ALN[3], &ALN[4], &ALN[5]);
200            fflush(stdin);
201            is_ok = (isCorrectLotteryRow(ALN));
202            first_input = false;
203
204        } while(is_ok == false);
205
206        first_input = true;     // reset to true for next evaluation
207
208        // Actual Lottery Super Number '''''''''''''''''''''''''''''''''''''''''''''
209
210        do {
211            if(first_input == true)
212                printf("Enter actual super number: ");
213            else
214                printf("Invalid input! Please correct: ");
215            scanf("%i", &ASN);
216            fflush(stdin);
217            first_input = false;
218
219        } while(ASN < 0 || ASN > 9);
220
221        first_input = true;     // reset to true for next evaluation
222
223        // Actual Game 77 ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
224
225        if(current.T_G77[0] == 'y' ){
226
227            do {
228                if(first_input == true)
229                    printf("Enter actual Game 77 number: ");
230                else
231                    printf("Invalid input! Please correct: ");
232                scanf("%7[0123456789]", cG77);
233                fflush(stdin);
234                first_input = false;
235
236            } while((strlen(cG77)) < 7);
237
238            first_input = true;     // reset to true for next evaluation
239        }
240
241        // Actual Super 6 ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
242
243        if(current.T_SU6[0] == 'y'){
244
245            do {
246                if(first_input == true)
247                    printf("Enter actual Super 6 number: ");
248                else
249                    printf("Invalid input! Please correct: ");
250            scanf("%6[0123456789]", cSU6);
251            fflush(stdin);
252            first_input = false;
253
254            } while((strlen(cSU6)) < 6);
255
256            first_input = true;     // reset to true for next evaluation
257        }
258
259        // Actual Glueckspirale '''''''''''''''''''''''''''''''''''''''''''''''''''''''
260
261
262
263
```

```c
264        // Console Output
265
266        strftime(sDrwDate, N, "%A, %d-%b-%Y", &dd);
267
268        printf("\nCheck your input:\n");
269        printf("\nDrawing Date: %s\n", sDrwDate);
270         printf("Actual lottery numbers: %i %i %i %i %i %i\n",ALN[0],ALN[1],ALN[2],ALN[3],ALN[4],ALN[5]);
271         printf("Actual super number: %i\n", ASN);
272        if(current.T_G77[0] == 'y')
273            printf("Actual Game 77: %s\n", cG77);
274        else
275            printf("Actual Game 77: N/A\n");
276        if(current.T_SU6[0] == 'y')
277             printf("Actual Super 6: %s\n", cSU6);
278        else
279            printf("Actual Super 6: N/A\n");
280         return 0;
281
282    }
283
284    /****************************************************************************
285     * check Lotto
286     ****************************************************************************
287
288     Evaluates actual lottery numbers against lottery ticket
289     in order to determine matches and win classes                          */
290
291
292
293    int checkLotto() {
294
295        int RowNo, i, j;                          // row number, indizees
296        int MPR[NOLR];                              // matches per lottery row
297        int WinRows = 0;                          // number of lottery rows with win
298        bool CSN;                                 // indicates correct super number matching last digit of the ticket number
299        char sDrwDate[N];                         // Drawing date formated
300        int iNOLR = current.T_Max_Row;            // Number of lottery rows active on the ticket
301        char WinMsg[12];                          // literal to show win class for ouput
302
303        // Initialize MPR and CSN
304
305        for(i = 0; i < NOLR; i++) {
306           MPR[i] = 0;
307        }
308
309        CSN = false;
310
311    // check for matches with actual lottery numbers '''''''''''''''''''''''''''''''
312
313        for(RowNo = 0;RowNo < NOLR; RowNo++)
314            for(i = 0; i < 6; i++)
315                for(j = 0; j < 6; j++)
316                    if(current.T_Row[RowNo][i] == ALN[j])
317                        MPR[RowNo]++ ;
318
319        // check for correct Super Number '''''''''''''''''''''''''''''''''''''''''
320
321        if(ASN == convertToDigit(current.T_No[6]))
322                    CSN = true;
323
324        // Generate output ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
325
326        // Console output + result file output for lottery results
327
328        strftime(sDrwDate, N, "%A, %d-%b-%Y", &dd);
329        printf("\nLottery Matches on %s\n\n", sDrwDate);
```

```
330
331         // Loop through lottery rows to generate console and file output for matches and win(s)
332
333         for(RowNo = 0; RowNo < NOLR; RowNo++) {
334
335             strcpy(WinMsg, getWinClass(MPR[RowNo], CSN));
336
337             if(strcmp(WinMsg, "no win") != 0)
338                 WinRows++; //counting number of rows with win
339
340             switch(MPR[RowNo]) {
341
342                 case 0:
343                     printf("Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / no match)\n",RowNo + 1,current.T_Row[RowNo][0], current.
344                     fprintf(pFile, "Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / no match)\n",RowNo + 1,current.T_Row[RowNo][0],
345                 break;
346
347                 case 1:
348                     printf("Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i match)\n",RowNo + 1,current.T_Row[RowNo][0], current.
349                     fprintf(pFile, "Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i match)\n",RowNo + 1,current.T_Row[RowNo][0],
350                 break;
351                 default:
352                     if(CSN == true) {
353                         printf("Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i matches + correct super number)\n",RowNo + 1,curr
354                         fprintf(pFile, "Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i matches + correct super number)\n",RowNo
355                     } else {
356                         printf("Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i Matches)\n",RowNo + 1,current.T_Row[RowNo][0], cu
357                         fprintf(pFile, "Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i Matches)\n",RowNo + 1,current.T_Row[RowNo]
358                     }
359                 break;
360             }
361         }
362
363
364         switch(WinRows) {
365
366             case 0: {
367                 if(iNOLR == 1) {
368                     printf("\nThere is no win for any of %i row played in total.\n",iNOLR);
369                     fprintf(pFile, "\nThere is no win for any of %i row played in total.\n",iNOLR);
370                 } else {
371                     printf("\nThere is no win for any of %i rows played in total.\n",iNOLR);
372                     fprintf(pFile, "\nThere is no win for any of %i rows played in total.\n",iNOLR);
373                 }
374                 break;
375             }
376             case 1: {
377                 if(iNOLR == 1) {
378                     printf("\nThere is %i row with win of %i row played in total.\n",WinRows,iNOLR);
379                     fprintf(pFile, "\nThere is %i row with win of %i row played in total.\n",WinRows,iNOLR);
380                 } else {
381                     printf("\nThere is %i row with win of %i rows played in total.\n",WinRows,iNOLR);
382                     fprintf(pFile, "\nThere is %i row with win of %i rows played in total.\n",WinRows,iNOLR);
383                 }
384                 break;
385             }
386             default: {
387                 printf("\nThere are %i rows with wins of %i row(s) played in total.\n",WinRows,iNOLR);
388             }
389         }
390
391         return 0;
392     }
393
394     /*****************************************************************************
395      * check Game 77
```

```
396     **************************************************************************

397
398     Evaluates actual lottery numbers against lottery ticket
399     in order to determine matches and win classes                            */

400
401     int checkGame77() {

402
403         int ii;                                      // index of ticket number array
404         int MatchG77 = 0;                            // matches Game 77
405         char WinClassG77[4];                         // Game 77 win class

406

407
408         if(current.T_G77[0] == 'y' ){

409
410             for(ii = 6; ii >= 0; ii--) {
411                 if(current.T_No[ii] == cG77[ii])
412                     MatchG77++;
413                 else
414                 break;
415             }

416
417             if(MatchG77 > 0) {
418                 switch(MatchG77) {

419
420                     case 1: strcpy(WinClassG77, "VII"); break;
421                     case 2: strcpy(WinClassG77, "VI"); break;
422                     case 3: strcpy(WinClassG77, "V"); break;
423                     case 4: strcpy(WinClassG77, "IV"); break;
424                     case 5: strcpy(WinClassG77, "III"); break;
425                     case 6: strcpy(WinClassG77, "II"); break;
426                     case 7: strcpy(WinClassG77, "I"); break;
427                     default: strcpy(WinClassG77, "---");
428                 }

429
430                 if(MatchG77 == 1) {
431                     printf("You have won Game 77 according winning class %s (%i digit matching).\n",WinClassG77,MatchG77);
432                     fprintf(pFile, "You have won Game 77 according winning class %s (%i digit matching).\n",WinClassG77,MatchG7
433                 }

434
435                 if(MatchG77 > 1) {
436                     printf("You have won Game 77 according winning class %s (%i digits matching).\n",WinClassG77,MatchG77);
437                     fprintf(pFile, "You have won Game 77 according winning class %s (%i digits matching).\n",WinClassG77,MatchG
438                 }

439
440             } else {
441                 printf("There is no win for Game 77.\n");
442                 fprintf(pFile, "There is no win for Game 77.\n");

443
444             }
445         } else {

446
447             printf("Game 77 is not appicable for this ticket.\n");
448             fprintf(pFile, "Game 77 is not appicable for this ticket.\n");
449         }

450
451         return 0;
452     }

453
454     /**************************************************************************
455      * check Super 6
456      **************************************************************************

457
458     Evaluates actual lottery numbers against lottery ticket
459     in order to determine matches and win classes                            */

460
461     int checkSuper6() {
```

```c
462
463
464        int ii;                                          // index of ticket number array
465        int MatchSU6 = 0;                                // matches Super 6
466        char WinClassSU6[4];                             // Game 77 win class
467
468     if(current.T_SU6[0] == 'y'){
469
470         for(ii = 6; ii >= 0; ii--) {
471         if(current.T_No[ii] == cSU6[ii -1])
472             MatchSU6++;
473         else
474             break;
475     }
476
477         if(MatchSU6 > 0) {
478             switch(MatchSU6) {
479
480                 case 1: strcpy(WinClassSU6, "VI"); break;
481                 case 2: strcpy(WinClassSU6, "V"); break;
482                 case 3: strcpy(WinClassSU6, "IV"); break;
483                 case 4: strcpy(WinClassSU6, "III"); break;
484                 case 5: strcpy(WinClassSU6, "II"); break;
485                 case 6: strcpy(WinClassSU6, "I"); break;
486                 default: strcpy(WinClassSU6, "---");
487             }
488
489
490             if(MatchSU6 == 1) {
491                 printf("You have won Super 6 according winning class %s (%i digit matching).\n",WinClassSU6,MatchSU6);
492                 fprintf(pFile, "You have won Super 6 according winning class %s (%i digit matching).\n",WinClassSU6,MatchSU
493             }
494
495             if(MatchSU6 > 1) {
496                 printf("You have won Super 6 according winning class %s (%i digits matching).\n",WinClassSU6,MatchSU6);
497                 fprintf(pFile, "You have won Super 6 according winning class %s (%i digits matching).\n",WinClassSU6,MatchS
498
499             }
500
501         } else {
502             printf("There is no win for Super 6.\n");
503             fprintf(pFile, "There is no win for Super 6.\n");
504
505         }
506     } else {
507
508         printf("Super 6 is not appicable for this ticket.\n");
509         fprintf(pFile, "Super 6 is not appicable for this ticket.\n");
510
511     }
512
513     return 0;
514 }
515
516 /***************************************************************************
517  * check GLUCKSPIRALE
518  ***************************************************************************
519
520 Evaluates ticket against actual result for 'Glueckspirale'. */
521
522 // TODO (camelo#1#01/03/18): Implement evaluation for Glueckspirale
523
524 int checkGSP() {
525
526
527     return 0;
```

```
528    }
```

t_select.c

```c
/*t_select.c | RLotto | gcc | v0.8.354.1715
 * Console program for storing and evaluating lottery ticket results.
 * ----------------------------------------------------------------------------
 *
 * Objective:    Select stored ticket for evaluation
 *
 * Author:        Reinhard Rozumek
 * Email:         reinhard@rozumek.de
 * Created:       10/08/17
 * Last mod:      12/28/17
 *
 * ----------------------------------------------------------------------------
 * This file is part of RLotto.                                              */

 // HEADER SECTION

#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <dirent.h>
#include "rlotto.h"


/* FUCTION DECLARATION ***************************************************/

void read_Ticket(char *ticket_no);
void display_Ticket(void);


/*    SELECT TICKET *********************************************************
     Search path of executable binary for lottery tickets and displays a sub menu
     to select which ticket to open. Afterwards read_Ticket function is called to
     read the ticket data from file into global variables. Finally and after user
     confirms selected ticket the function for ticket evaluation is called.
     *********************************************************************/


int selectTicket(void) {

    char ticket_no[7];

    printf("\nTICKET SELECTION\n\n");

    // Find all files in path of binary executable file. Requires dirent.h

    int i = 0;
    int i_min = 1, i_max;
    int i_input = 0;
    DIR *d;
    struct dirent *dir;
    char fn_array[100] [12];                             // Array filename with extension (str length + 1)

    // First run just to count number of tickets
    d = opendir(TicketFolder);

    if (d)
    {
        while ((dir = readdir(d)) != NULL)
        {
            char *fn = dir->d_name;                      // fn: file name
            char * ext = strrchr(fn, '.');               // ext: file extension
            if(strcmp(ext, T_EXT) == 0)                  // File extension for lottery tickets
            {
```

```
 66                      i++;
 67                  }
 68              }
 69
 70          closedir(d);
 71          i_max = i;
 72      }
 73
 74
 75
 76      // Second run to read ticket file names in array for all files matching file extension
 77      d = opendir(TicketFolder);
 78
 79      if (d)
 80      {
 81          i = 0;                                      // Reset counter i
 82          while ((dir = readdir(d)) != NULL)
 83          {
 84              char *fn = dir->d_name;                 // fn: file name
 85              char * ext = strrchr(fn, '.');          // ext: file extension
 86              if(strcmp(ext, T_EXT) == 0)             // File extension for lottery tickets
 87              {
 88                  i++;
 89                  strcpy(fn_array[i-1],fn);
 90                  printf("%3i --> Ticket Number: %.*s File Name: %s\n", i, 7, fn + 0, fn_array[i-1]);
 91              }
 92          }
 93          closedir(d);
 94      }
 95
 96
 97      // i equals null --> no ticket found
 98      if(i == 0)
 99      {
100          // no ticket found
101          printf("\nNo ticket found in selected directory.\n");
102      }
103      else
104      {
105          printf("\n%i ticket(s) found in selected directory.\n", i_max);
106          do {
107              printf("\nEnter number between %i and %i: ",i_min, i_max);
108              scanf("%d", &i_input);
109              if(i_input < 1 || i_input > i_max)
110              if(i_input < 1 || i_input > i_max)
111              {
112                  printf("\n \"%d\" is not between %i and %i !\n",i_input, i_min, i_max);
113                  printf("\n \"%d\" is not between %i and %i !\n",i_input, i_min, i_max);
114              }
115          }while(i_input < 1 || i_input > i_max);
116
117          strncpy(ticket_no, fn_array[i_input - 1] + 0,7);
118          ticket_no[7] = '\0';
119          printf("\nTicket No %s has been selected.\n", ticket_no);
120      }
121
122
123
124      // Call function for reading selected ticket
125
126      read_Ticket(ticket_no);
127
128      return 0;
129  }
130
131  /*    READ TICKET *********************************************************
```

```
132          Reads selected ticket into ticket structure
133          ***********************************************************************/
134
135      void read_Ticket(char *ticket_no) {
136
137          FILE *fp;                           // file pointer
138          char line[MAX_LINE_LENGTH];          // array for line string
139          int lnr;                             // line number of input file
140          char *description;                   // left token of separated input line contains line description
141          char *value;                        // right token of separated input line contains line value
142          char *token1, *token2, *token3;     // token 1-3 for lottery numbers
143          char *token4, *token5, *token6;     // token 4-6 for lottery numbers
144          int sConfirm;                        // yes or no to confirm ticket for further evaluation
145
146
147          // construct filename
148          char t_filename[45];
149          strcpy(t_filename, TicketFolder);
150          strcat(t_filename, ticket_no);
151          strcat(t_filename, T_EXT);
152
153          // Open file for reading ---------------------------------
154
155          lnr = 0;
156          fp = fopen(t_filename, "r");
157
158          if(fp == NULL) {
159
160              printf("Error opening file!\n");
161
162          } else {
163
164              while(fgets(line, MAX_LINE_LENGTH, fp) != NULL){
165
166                  strtok(line, "\n");               // Remove trailing new line character
167
168                  lnr++;
169
170                  // Parsing input and copy into current ticket structure ----------------------------------
171
172                  // separate input line
173
174                  description = strtok(line, ":");
175                  value = strtok(0, ":");
176
177                  // reading values into ticket structure
178
179                  switch(lnr){
180
181                      case 1:
182                          strcpy(current.T_No, value); break;
183                      case 2:
184                          strcpy(current.T_Player, value); break;
185                      case 3:
186                          strcpy(current.T_Start, value); break;
187                      case 4:
188                          strcpy(current.T_Runtime, value); break;
189                      case 5:
190                          strcpy(current.T_D_Day, value); break;
191                      case 6:
192                          strcpy(current.T_G77, value); break;
193                      case 7:
194                          strcpy(current.T_SU6, value); break;
195                      case 8:
196                          strcpy(current.T_GSP, value); break;
197                      case 9:
```

```
198                         current.T_Max_Row = atoi(value); break;
199             case 10:
200
201                 token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
202                 token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
203                 token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
204                 token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
205                 token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
206                 token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
207                 break;
208
209             case 11:
210
211                 token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
212                 token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
213                 token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
214                 token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
215                 token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
216                 token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
217                 break;
218
219             case 12:
220
221                 token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
222                 token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
223                 token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
224                 token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
225                 token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
226                 token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
227                 break;
228
229             case 13:
230
231                 token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
232                 token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
233                 token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
234                 token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
235                 token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
236                 token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
237                 break;
238
239             case 14:
240
241                 token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
242                 token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
243                 token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
244                 token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
245                 token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
246                 token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
247                 break;
248
249             case 15:
250
251                 token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
252                 token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
253                 token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
254                 token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
255                 token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
256                 token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
257                 break;
258
259             case 16:
260
261                 token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
262                 token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
263                 token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
```

```
264                    token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
265                    token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
266                    token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
267                    break;
268
269            case 17:
270
271                    token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
272                    token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
273                    token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
274                    token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
275                    token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
276                    token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
277                    break;
278
279            case 18:
280
281                    token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
282                    token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
283                    token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
284                    token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
285                    token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
286                    token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
287                    break;
288
289            case 19:
290
291                    token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
292                    token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
293                    token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
294                    token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
295                    token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
296                    token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
297                    break;
298
299            case 20:
300
301                    token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
302                    token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
303                    token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
304                    token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
305                    token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
306                    token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
307                    break;
308
309            case 21:
310
311                    token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
312                    token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
313                    token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
314                    token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
315                    token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
316                    token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
317                    break;
318
319            default:
320            printf("\n Error parsing ticket file.\n");
321
322        }
323
324    }
325
326
327    fclose(fp);
328
329  }
```

```
330
331            // Display selected Ticket ---------------------------------

332
333            display_Ticket();

334
335            // confirm for evaluation ---------------------------------

336
337            printf("Evaluate this ticket now? [y/n]: ");

338
339            do {
340                sConfirm = tolower(mygetc());
341                if(sConfirm != 'y' && sConfirm != 'n')
342                    printf("\nEnter \"y\" or \"n\"\n");
343                if(sConfirm == 'y')
344                    evaluateTicket();
345                if(sConfirm == 'n') {
346                    printf("\nReturning to main menu.\n");
347                    t_initialize();
348                    // exit statement removed due to compiler warning (
349                }

350
351            } while(sConfirm != 'y' && sConfirm != 'n');

352
353        }
```

create_rlotto_tex_file.py

```python
#!/usr/bin/python
# read the *.c, *.h and *.py files from RLOTTO project and
# create a nice, beautified PDF document with it.
# The output of this script further processed by lualatex

import argparse
import datetime
now = datetime.datetime.now()


parser = argparse.ArgumentParser(description='Create PDF manual from source code')
parser.add_argument('--header',     help='*.h files', required=True)
parser.add_argument('--csource',     help='*.c files', required=True)
parser.add_argument('--python',     help='*.py files', required=False)

args = parser.parse_args()

LATEX1="""\\documentclass[a4paper,10pt]{scrartcl}
\\KOMAoptions{DIV=12}
\\parindent=0pt
\\usepackage{minted}

\\setminted[C]{linenos,fontsize=\\footnotesize,tabsize=4}
\\setminted[make]{linenos,fontsize=\\footnotesize,tabsize=4}
\\setminted[python]{linenos,fontsize=\\footnotesize,tabsize=4}

% the following is needed for syntax highlighting
\\usepackage{color,hyperref}

\\hypersetup{backref,hidelinks,
colorlinks=false}
\\definecolor{dkgreen}{rgb}{0,0.6,0}
\\definecolor{gray}{rgb}{0.5,0.5,0.5}
\\definecolor{mauve}{rgb}{0.58,0,0.82}
\\begin{document}
\\underline{\\Large{\\textbf{RLOTTO Code Manual}}}\\newline
"""

LATEX2="""\\bigskip
\\tableofcontents
"""

print(LATEX1)
print(str(now))
print(LATEX2)



#output Makefile
print("\\newpage\\section{{Makefile}}")
print("\\inputminted{make}{{../Makefile}}\\newpage")

# output *.h
print("\\newpage\\section{Header files}")
for f in args.header.split(" "):
    if f:
        t = f.replace("_","\\_")
        print("\\subsection*{\\texttt{" + t + "}}")
        print("\\addcontentsline{toc}{subsection}{\\texttt{" + t + "}}")
        print("\\inputminted{C}{../" + f + "}\\newpage")

# output *.c
print("\\section{Sourcecode}")
for f in args.csource.split(" "):
    if f:
```

```python
66              t = f.replace("_","\\_")
67              print("\\subsection*{\\texttt{" + t + "}}")
68              print("\\addcontentsline{toc}{subsection}{\\texttt{" + t + "}}")
69              print("\\inputminted{C}{../" + f + "}\\newpage")
70
71      if args.python:
72              t = args.python.replace("_","\\_")
73              print("\\subsection*{\\texttt{" + t + "}}")
74              print("\\addcontentsline{toc}{subsection}{\\texttt{" + t + "}}")
75              print("\\inputminted{python}{" + args.python + "}\\newpage")
76
77
78      # example ticket
79      print("\\section{Examples}")
80      print("\\subsection*{Ticket Input}")
81      print("\\addcontentsline{toc}{subsection}{Ticket Input}")
82      print("\\inputminted{text}{{../1234567.tck}}\\newpage")
83
84      # exmaple results
85      print("\\subsection*{Output Result}")
86      print("\\addcontentsline{toc}{subsection}{Output Result}")
87      print("\\inputminted{text}{{../results/Lotto-Result-2017-11-12.txt}}\\newpage")
88
89      print("""\\end{document}""")
```

4 Examples

Ticket Input

```
Ticket No:1234567
Player:Reinhard Rozumek
Play Date:19.11.2017
Runtime:1
Weekday:s
Game 77:y
Super 6:y
Glueckspirale:n
Active Rows:6
Row  1:4,6,11,13,17,31
Row  2:15,22,30,35,36,43
Row  3:9,22,28,34,43,47
Row  4:16,27,29,40,42,26
Row  5:16,25,39,40,44,48
Row  6:5,10,14,28,33,40
Row  7:0,0,0,0,0,0
Row  8:0,0,0,0,0,0
Row  9:0,0,0,0,0,0
Row 10:0,0,0,0,0,0
Row 11:0,0,0,0,0,0
Row 12:0,0,0,0,0,0
```

Output Result

```
RLOTTO v0.5.293.1523
Evaluating lottery results
Lottery Ticket No: 3124567


Players: Max Mustermann


Drawing Date: Sunday, 12-Nov-2017
Lottery numbers: 1 2 3 4 5 6
Bonus number: 8
Super number: 9
Game 77: 1234567
Super 6: 123456


Lottery Matches on Sunday, 12-Nov-2017


Row No  1: 38 40 41 42 44 47        (no win / no match)
Row No  2: 11 17 18 36 33 45        (no win / no match)
Row No  3: 19 20 30 41 46 48        (no win / no match)
Row No  4:  0  0  0  0  0  0        (no win / no match)
Row No  5:  0  0  0  0  0  0        (no win / no match)
Row No  6:  0  0  0  0  0  0        (no win / no match)
Row No  7:  0  0  0  0  0  0        (no win / no match)
Row No  8:  0  0  0  0  0  0        (no win / no match)
Row No  9:  0  0  0  0  0  0        (no win / no match)
Row No 10:  0  0  0  0  0  0        (no win / no match)
Row No 11:  0  0  0  0  0  0        (no win / no match)
Row No 12:  0  0  0  0  0  0        (no win / no match)


There is no win for any of 12 rows played in total.
You have won Game 77 according winning class IV (4 digits matching).
There is no win for Super 6.


Without any warranty.
```