RLOTTO Code Manual

2018-03-03 23:19:42.024611

Contents

1 Makefile

```
# compile instructions from: https://github.com/prozumr/RLOTTO2/issues/1
# gcc rlotto.c t_select.c t_add.c t_evaluate.c t_delete.c settings.c t_common.c -o rlotto.exe

CC        = gcc
CFLAGS    = -Wall
BIN       = rlotto.exe
RLOTTOTEX = rlotto-manual.tex
LATEX     = lualatex
LATEXOPT  = --interaction=batchmode -shell-escape

HEADER    = rlotto.h  version.h
CFILES    = rlotto.c  settings.c  t_add.c  t_common.c  t_delete.c \
            t_evaluate.c  t_select.c
PYTHON    = create_rlotto_tex_file.py
OBJ       = rlotto.o t_select.o t_add.o t_evaluate.o t_delete.o settings.o \
            t_common.o

all: $(OBJ)
    $(CC)  -DMYSTRING='"hello"' $(CFLAGS) -o $(BIN) $(OBJ)

.PHONY: manual
manual:
    cd manual; \
    ./create_rlotto_tex_file.py \
        --header  "$(HEADER)"   \
        --csource "$(CFILES)"   \
        --python  "$(PYTHON)"   \
        > $(RLOTTOTEX);         \
    $(LATEX) $(LATEXOPT) $(RLOTTOTEX); \
    $(LATEX) $(LATEXOPT) $(RLOTTOTEX)


.PHONY: clean
clean:
    rm -rf $(BIN) $(OBJ)
    cd manual && rm -rf *.aux *.log *.toc *.out manual/_minted*
```

2 Header files

rlotto.h

```
 /*rlotto.h | RLotto | gcc | v0.8.354.1715
 * Console program for storing and evaluating lottery ticket results.
 * ---------------------------------------------------------------------------
 *
 * Objective:    Providing global variables and functions for RLotto
 *
 * Author:         Reinhard Rozumek
 * Email:          reinhard@rozumek.de
 * Created:     10/08/17
 * Last mod:    02/05/18
 *
 * ---------------------------------------------------------------------------
 * This file is part of RLotto.
 *
 * c 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 RLOTTO2. If not, see <http://www.gnu.org/licenses/>.
 */



#ifndef RLOTTO_H
#define RLOTTO_H
#define N 80
#define NOLR 12                                                                    // N_umber O_f L_ottery R_ows -1 (First Row with

 /* GLOBAL VARIABLES */
bool go_Exit;
struct date { short month, day, year; };                                        // not used !? => check

extern int ALN[6];                                                                // Actual Lottery Numbers
extern int ASN;                                                                    // Actual Super Number
extern char cG77[8];                                                            // Actual Game 77 Number
extern char cSU6[7];                                                            // Actual Super 6 Number
extern char TicketFolder[13];                                                     // Folder to store results
extern char ResultFolder[13];                                                     // Folder to store results
extern FILE *pFile;                                                             // Pointer to result file

/* GLOBAL DEFINITIONS */
#define MAX_T_PLAYER_SZ 64                                // Max characters for player name
#define MAX_LINE_LENGTH 80                                // Max characters for length of a ticket row
#define T_EXT ".tck"                                    // File extension for lottery tickets




 /* FUNCTION DECLARATIONS */

    int getche(void);
    int terminateProgram(void);
    int welcome(void);
    int selectTicket(void);
    int addTicket(void);
```

```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

```
 /*version.h | RLotto | gcc | v0.8.354.1715
 * Console program for storing and evaluating lottery ticket results.
 * ---------------------------------------------------------------------------
 *
 * Objective:    maintain version information for source code
 *
 * Author:          Reinhard Rozumek
 * Email:          reinhard@rozumek.de
 * Created:      10/08/17
 * Last mod:     02/11/18
 *
 * ---------------------------------------------------------------------------
 * This file is part of RLotto.                                              */


#ifndef VERSION_H
#define VERSION_H

    // TODO (camelo#3#01/03/18): Improve handling of version number

    //Program Name
    static const char THISPROG[] = "RLOTTO";

    //Software Status
    static const char STATUS[] = "v0.8 beta";
    static const char STATUS_SHORT[] = "v0.8b";

    //Standard Version Type
    static const long MAJOR = 0;
    static const long MINOR = 8;
    static const long BUILD = 354;
    static const long REVISION = 1715;

    //Miscellaneous Version Types
    static const long BUILDS_COUNT = 354;
    #define RC_FILEVERSION 0,8,353,1715
    #define RC_FILEVERSION_STRING "0, 8, 354, 1715\0"
    static const char FULLVERSION_STRING[] = "0.8.354.1715";

    //These values are to keep track of your versioning state, don't modify them.
    static const long BUILD_HISTORY = 29;


#endif //VERSION_h



/*

ABSTRACT

Originally generated by a Code::Blocks plugin. RLOTTO continues to use version.h
for central maintenance of versioning. However and since RLOTTO is developed with
no real IDE but just an editor (VIM/GVIM/Notepad++), FAR and the GNU Compiler this
file is maintained manually.


VERSION VALUES

Major - Increments by 1 when the minor version reaches its maximum
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
Build number (also equivalent to Release) - Increments by 1 every time that the revision number is incremented.
Revision - Increments randomly when the project has been modified and then compiled.

STATUS
```

```

Some fields to keep track of your software status with a list of predefined values for convenience.

Software Status - The typical example should be v1.0 Alpha
Abbreviation - Same as software status but like this: v1.0a

SCHEME

Minor maximum - The maximum number that the Minor value can reach, after this value is reached the Major is incremented by 1
Build Number maximum - When the value is reached, the next time the project is compiled is set to 0. Put a 0 for unlimited.
Revision maximum - Same as Build Number maximum. Put a 0 for unlimited
Revision random maximum - The revision increments by random numbers that you decide, if you put here 1, the revision obvious
Build times before incrementing Minor - After successful changes to code and compilation the build history will increment, an
 */
```

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");
```

```c
    return(ch);
}
#endif

/******************************************************************************
 * MAIN
 ******************************************************************************/

/* Function main just displays the main user menu and generates the main
    program loop. Program can only be terminated by the option from the main
    menu. All submenus are coded in other functions. */

int main() {

    int iSelect;                                              // Selection from Main Menu

    welcome();                                                  // Welcome Message
    t_initialize();                                             // Initialize ticket data structure

    // TODO (camelo#3#01/03/18): Implement Console attribute (Title, Color etc.) ...

    /* Main Loop starts here */

    do {

        printf("\nMain Menu Options\n");
        printf("------------------------------------\n");
        printf("\t1 - Add Lottery Ticket\n");
        printf("\t2 - Delete Lottery Ticket\n");
        printf("\t3 - Evaluate Lottery Ticket\n");
        printf("\t4 - RLotto settings\n");
        printf("\t5 - Terminate Program\n");

        do {

            printf("\nPlease select (1-5): ");
            iSelect =(mygetc());
            fflush(stdin);

         } while(iSelect < 49 || iSelect > 53);

        iSelect = iSelect - 48;

        switch(iSelect) {

            case 1: addTicket(); break;
            case 2: deleteTicket(); break;
            case 3: selectTicket(); break;
            case 4: configureSettings(); break;
            case 5: terminateProgram();

        }

        } while(go_Exit == false);

}

/******************************************************************************
 * WELCOME
 ******************************************************************************/

 /* Welcome message and check for default RLotto file to open. */

int welcome(void) {

    printf("\n%s v%ld.%ld.%ld.%ld\n",THISPROG,MAJOR,MINOR,BUILD,REVISION);
    printf("Evaluating lottery results\n") ;

    return 0;
}

/******************************************************************************
 * INITIALIZE TICKET STRUCTURE
 ******************************************************************************/

/* Initializing all members of the ticket data structure.*/

int t_initialize(void) {

    // Initialize ticket structure
    strcpy(current.T_No, "0000000");
    strcpy(current.T_Player, "0000000000000000000000000000000000000000000000000000000000000000");
    strcpy(current.T_Start, "00/00/0000");
    strcpy(current.T_Runtime, "0");
    strcpy(current.T_D_Day, "u");
    strcpy(current.T_G77, "u");
    strcpy(current.T_SU6, "u");
    strcpy(current.T_GSP, "u");
    current.T_Max_Row = 0;

    for(int j = 0; j < 12; j++) {
        for(int k = 0; k < 6; k++) {
            current.T_Row[j][k] = 0;
        }
    }

    strcpy(display_runtime, "000000000");
    strcpy(display_weekday, "000000000");
    strcpy(display_G77, "000");
    strcpy(display_SU6, "000");
    strcpy(display_GSP, "000");

    return 0;

}


/******************************************************************************
 * TERMINATE PROGRAM
 ******************************************************************************/

/* terminateProgram displays a message for the program termination. Other
    functions like closing any open files, reset variables and write a history
     message to the ini file still needs to be defined. */

int terminateProgram(void) {

    printf("\n");
     printf("\n\nProgram terminated by user.\n");
     go_Exit = true; // remove after Main Menu is completed
     return 0;
}
```

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 {

        if(first_input == true)
            printf("\n\nAdd new lottery ticket? (y/n): ");
        else
            printf("Invalid input! Please correct: ");
        scanf("%s", sConfirm);
        fflush(stdin);

        if(*sConfirm=='y' || *sConfirm=='n') {
            is_ok = true;
        }

        first_input = false;

    } while(is_ok == false);

    // INPUT SUBMENUS - EACH INPUT QUERY ENCAPSULATED IN ITS OWN FUNCTION
    if(*sConfirm=='y') {

        t_initialize();
        get_ticket_No();
        get_Player_Name();
        get_Start_Date();
        get_T_Runtime();
        get_T_D_Day();
        get_T_G77();
        get_T_SU6();
        get_T_GSP();
        get_T_Rows();
        display_Ticket();

        // CONFIRM TO WRITE TICKET
        is_ok = false;
        first_input = true;

        do {

            if(first_input == true)
                printf("\nWrite ticket to file sytem? (y/n): ");
            else
                printf("Invalid input! Please correct: ");
            scanf("%s", sWrite);
            fflush(stdin);

            if(*sWrite=='y' || *sWrite=='n') {
                is_ok = true;
            }

            first_input = false;

        } while(is_ok == false);

        // WRITE TICKET TO FILE SYSTEM
        if(*sWrite=='y') {

            write_Ticket();

        } else {

            printf("\nCreation of new ticket canceled.\n");
        }

    } else {

        printf("\nCreation of new ticket canceled.\n");
    }

```

```c
    return 0;

}

/*****************************************************************************
 * GET TICKET NUMBER
 ****************************************************************************/

 /*  Gets Ticket Number by user input from stdin. */


 void get_ticket_No(void) {

     bool is_ok = false;            // correctness of input format
     bool first_input = true;        // indicates first attempt for input
     char ticket_No[8];                // Ticket No to return to calling function

     do
    {
        if(first_input == true)
            printf("Enter 7-digit Ticket Number: ");
        else
            printf("Invalid input! Please correct: ");
        scanf("%8s", ticket_No);                          //Input limited to 7 characters
        fflush(stdin);
        is_ok = (isCorrectTicketNo(ticket_No));
        first_input = false;

    } while(is_ok == false);

    /* Remove trailing newline, if there and add 0 terminator. */
    if ((strlen(ticket_No)>0) && (ticket_No[strlen (ticket_No) - 1] == '\n'))
        ticket_No[strlen (ticket_No) - 1] = '\0';

    strcpy(current.T_No , ticket_No);                        // Assign ticket number to structure

 }

 /****************************************************************************
 * GET PLAYER NAME(S)
 ****************************************************************************/

 /*  Gets Player Name by user input from stdin. */

 void get_Player_Name(void) {

    char cPlayers[MAX_T_PLAYER_SZ];

    printf("Enter Player Name: ");
    fgets (cPlayers, MAX_T_PLAYER_SZ, stdin);

    /* Remove trailing newline, if there and add 0 terminator. */
    if ((strlen(cPlayers)>0) && (cPlayers[strlen (cPlayers) - 1] == '\n'))
        cPlayers[strlen (cPlayers) - 1] = '\0';

    strcpy(current.T_Player, cPlayers);                    // Assign ticket number to structure

 }

 /****************************************************************************
 * GET TICKET START DATE
 *****************************************************************************/

/*  Gets Date of purchase by user input from stdin. */

 void get_Start_Date(void) {

    struct tm ts;                // date type for t_icket s_tart date

    int year, month, day;        // year, month, day as enterd by user
    bool is_ok = false;            // correctness of date format
    char sPlayDate[11];            // Actual Drawing Date
    bool first_input = true;    // indicates first attempt for input

    do
    {
        if(first_input == true)
            printf("Enter playing date (mm/dd/yyyy): ");
        else
            printf("Invalid input! Please correct: ");
        scanf("%d/%d/%d", &month, &day, &year);
        fflush(stdin);
        is_ok = (isCorrectDateFormat(month, day, year));
        first_input = false;

    } while(is_ok == false);

    ts.tm_year = year - 1900;
    ts.tm_mon  = month - 1;
    ts.tm_mday = day;

    ts.tm_hour = 0;
    ts.tm_min  = 0;
    ts.tm_sec  = 1;
    ts.tm_isdst = -1;

    if ( mktime(&ts) == -1 )
      ts.tm_wday = 7;

    // strftime(sPlayDate, 40, "%A, %d-%B-%Y", &ts); <----

    strftime(sPlayDate, 11, "%d.%m.%Y", &ts);


    strcpy(current.T_Start , sPlayDate);                        // Assign ticket Start Date to structure

    is_ok = false;        // reset to false for next evaluation
    first_input = true;    // reset to true for next evaluation

 }

 /******************************************************************************
 * GET TICKET RUNTIME
 ******************************************************************************/

 /* Gets ticket runtime in weeks by user input from stdin.
    Allowed ranged is 1-5 (weeks),'m' (month) or 'p' (permanent) */


 void get_T_Runtime(void) {

    char rt[2];                    // Ticket runtimet in weeks .
    bool is_ok = false;            // correctness of date format
    bool first_input = true;    // indicates first attempt for input

    do
    {
        if(first_input == true)
            printf("Enter ticket runtime (1-5,m,p): ");
        else
            printf("Invalid input! Please correct: ");
        scanf("%s", rt);
```

```c
        fflush(stdin);
        if(*rt=='1' || *rt=='2' || *rt=='3' || *rt=='4' || *rt=='5' || *rt=='m' || *rt=='p' ) {

            is_ok = true;

            /* Remove trailing newline, if there and add 0 terminator. */
            if ((strlen(rt)>0) && (rt[strlen (rt) - 1] == '\n'))
                rt[strlen (rt) - 1] = '\0';

            strcpy(current.T_Runtime, rt);                        // Assign ticket runtime to structure
        }

        first_input = false;

    } while(is_ok == false);
 }

/*****************************************************************************
 * GET TICKET DRAWING DATE OF THE WEEK
 *****************************************************************************/

 /* Gets ticket drawing day of the week by user input from stdin.
    Allowed ranged is s (Saturday), w (Wednesday) or b (Both - Saturday and
    Wednesday) */

void get_T_D_Day(void){

    char T_D_Day[2];            // Ticket runtimet in weeks .
    bool is_ok = false;            // correctness of date format
    bool first_input = true;    // indicates first attempt for input

    do
    {
        if(first_input == true)
            printf("Enter ticket drawing day (s,w,b): ");
        else
            printf("Invalid input! Please correct: ");
        scanf("%s", T_D_Day);
        fflush(stdin);
        if(*T_D_Day=='s' || *T_D_Day=='w' || *T_D_Day=='b') {

            is_ok = true;

            /* Remove trailing newline, if there and add 0 terminator. */
            if ((strlen(T_D_Day)>0) && (T_D_Day[strlen (T_D_Day) - 1] == '\n'))
                T_D_Day[strlen (T_D_Day) - 1] = '\0';

            strcpy(current.T_D_Day, T_D_Day);                    // Assign ticket drawing day to structure
        }

        first_input = false;

    } while(is_ok == false);
}

/*****************************************************************************
 * GET GAME 77
 *****************************************************************************/

 /* Gets info by user if ticket is enabled for "Game 77" (German "Spiel 77").
    Allowed ranged is y (yes) or n (no) */

void get_T_G77(void){

    char T_G77[2];                // Indicates if Game 77 is active (y/n).
    bool is_ok = false;            // Indicates correctness of date format
    bool first_input = true;    // indicates first attempt for input

    do
    {
        if(first_input == true)
            printf("Enter if Game 77 is active (y,n): ");
        else
            printf("Invalid input! Please correct: ");
        scanf("%s", T_G77);
        fflush(stdin);
        if(*T_G77=='y' || *T_G77=='n') {

            is_ok = true;

            /* Remove trailing newline, if there and add 0 terminator. */
            if ((strlen(T_G77)>0) && (T_G77[strlen (T_G77) - 1] == '\n'))
                T_G77[strlen (T_G77) - 1] = '\0';

            strcpy(current.T_G77, T_G77);                        // Assign Game 77 (yes or no) to structure
        }

        first_input = false;

    } while(is_ok == false);
}

/******************************************************************************
 * GET SUPER 6
 ******************************************************************************/

/* Gets info by user if ticket is enabled for "Super 6".
    Allowed ranged is y (yes) or n (no) */

void get_T_SU6(void){

    char T_SU6[2];                // Indicates if Super 6 is active (y/n).
    bool is_ok = false;            // Indicates correctness of user input
    bool first_input = true;    // indicates first attempt for input

    do
    {
        if(first_input == true)
            printf("Enter if Super 6 is active (y,n): ");
        else
            printf("Invalid input! Please correct: ");
        scanf("%s", T_SU6);
        fflush(stdin);
        if(*T_SU6=='y' || *T_SU6=='n') {

            is_ok = true;

            /* Remove trailing newline, if there and add 0 terminator. */
            if ((strlen(T_SU6)>0) && (T_SU6[strlen (T_SU6) - 1] == '\n'))
                T_SU6[strlen (T_SU6) - 1] = '\0';

            strcpy(current.T_SU6, T_SU6);                        // Assign Super 6 (yes or no) to structure
        }

        first_input = false;

    } while(is_ok == false);
}

/******************************************************************************
 * GET GLUECKSPIRALE
 ******************************************************************************/
```

```c

/* Gets info by user if ticket is enabled for "Glueckspirale".
    Allowed ranged is y (yes) or n (no) */

void get_T_GSP(void){

    char T_GSP[2];                  // Indicates if Glueckspirale is active (y/n).
    bool is_ok = false;              // Indicates correctness of user input
    bool first_input = true;     // indicates first attempt for input

    do
    {
        if(first_input == true)
            printf("Enter if Glueckspirale is active (y,n): ");
        else
            printf("Invalid input! Please correct: ");
        scanf("%s", T_GSP);
        fflush(stdin);
        if(*T_GSP=='y' || *T_GSP=='n') {

            is_ok = true;

            /* Remove trailing newline, if there and add 0 terminator. */
            if ((strlen(T_GSP)>0) && (T_GSP[strlen (T_GSP) - 1] == '\n'))
                T_GSP[strlen (T_GSP) - 1] = '\0';

            strcpy(current.T_GSP, T_GSP);                        // Assign Glueckspirale (yes or no) to structure
        }

        first_input = false;

    } while(is_ok == false);

}

/*******************************************************************************
 * GET LOTTERY NUMBERS
 *******************************************************************************/

/* Get lottery numbers per row. 12 rows defined. Range of number is 1-49.*/

void get_T_Rows(void){

    int T_Max_Row;                  // Max number of rows used in this ticket
    int i, j;                      // Loop counter
    int checksum;                  // Checksum to validate numbers are in range of 1-49
    int countInput;                  // Used as count for input arguments of scanf per iteration
    int n[12][6];                  // Lottery number array
    bool is_ok = false;              // Indicates correctness of user input
    bool first_input = true;     // indicates first attempt for input

    // Query number of active rows (T_Max_Row)

    do
    {
        if(first_input == true)
            printf("Enter number of active rows (1-12): ");
        else
            printf("Invalid input! Please correct: ");
        scanf("%d", &T_Max_Row);
        fflush(stdin);
        if(T_Max_Row > 0 && T_Max_Row < 13) {

            is_ok = true;
            current.T_Max_Row = T_Max_Row;                        // Assign Glueckspirale (yes or no) to structure
        }

        first_input = false;

    } while(is_ok == false);

    // Query Lottery Numbers (T_Row[][])

    is_ok = false;                                                   // reset loop control variables
    first_input = true;                                               // reset loop control variables

    do
    {
        // Initialzing variables required to reset for each loop;
        checksum = 0;

        for(i = 0; i < 12; i++){
            for(j = 0; j < 6; j++){
                n[j][j]=0;
            }
        }

        if(first_input == true)
            printf("Enter Lottery numbers per row separated by commas.\n");
        else {
            printf("Invalid input! Please correct!\n\n");
        }


        for(i = 0; i <= T_Max_Row - 1; i++) {

            countInput = 0;

            printf("Row %d: ", i + 1);
            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]);
            fflush(stdin);

            if(countInput == 6){

                // Checks range of each mumber in the row
                for(j = 0; j < 6; j++){

                    if(n[i][j] > 0 && n[i][j] < 50){

                        checksum++;
                        current.T_Row[i][j] = n[i][j];
                    }
                }

            }else {
                printf("\nInvalid number of arguments in input\n");
                checksum = 0;
                t_initialize();                                       //re-initialie to ensure structure is clean
                break;
            }

            if(countInput == 6 && checksum != (i+1) * 6){
                printf("\nInvalid value - All lottery numbers have to be in range from 1 to 49 \n");
                checksum = 0;
                t_initialize();                                       //re-initialize to ensure structure is clean
                break;
            }
        }

        if(checksum == 6 * T_Max_Row){
            is_ok = true;
        }

        first_input = false;

    } while(is_ok == false);
}


/****************************************************************************
 * WRITE INPUT TO TICKET FILE
 ****************************************************************************/



void write_Ticket(void){

    // construct filename
    char t_full_path[45];
    strcpy(t_full_path, TicketFolder);
    strcat(t_full_path, current.T_No);
    strcat(t_full_path, T_EXT);

    printf("\nFull path: %s\n", t_full_path);


    // Open file and write
    FILE *fp;
    fp = fopen(t_full_path, "w");

    if(fp == NULL) {

        printf("\nTicket folder missing. Try to create now...\n");
            system("mkdir tickets");
            pFile = fopen(t_full_path, "w");
            if(pFile == NULL) {
                printf("Error opening %s for writing. Program terminated.", TicketFolder);
                abort();
            } else {
                printf("Folder \"%s\" has been created.\n", TicketFolder);
            }


    } else {

        fprintf(fp, "Ticket No:%s\n", current.T_No);
        fprintf(fp, "Player:%s\n", current.T_Player);
        fprintf(fp, "Play Date:%s\n", current.T_Start);
        fprintf(fp, "Runtime:%s\n", current.T_Runtime);
        fprintf(fp, "Weekday:%s\n", current.T_D_Day);
        fprintf(fp, "Game 77:%s\n", current.T_G77);
        fprintf(fp, "Super 6:%s\n", current.T_SU6);
        fprintf(fp, "Glueckspirale:%s\n", current.T_GSP);
        fprintf(fp, "Active Rows:%d\n", current.T_Max_Row);

        for(int j = 0; j < 12; j++) {

            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
        }

        fclose(fp);

        printf("\nTicket %s written.\n", t_full_path);
    }

}
```

t_common.c

```c
 /*t_common.c | RLotto | gcc | v0.8.354.1715
 * Console program for storing and evaluating lottery ticket results.
 * ---------------------------------------------------------------------------
 *
 * Objective:    Common functions used in various parts of this program.
 *
 * Author:          Reinhard Rozumek
 * Email:          reinhard@rozumek.de
 * Created:      11/17/17
 * Last mod:    02/04/18
 *
 * ---------------------------------------------------------------------------
 * 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"

/* GLOABL VARIABLES ***********************************************************/

struct tm dd;                                         // date type used for d_rawing d_ate


// FUNCTION DECLARATION


bool isCorrectTicketNo(char *n);
bool isCorrectDateFormat(int m, int d, int y);
bool isLeapYear(const int iYear);
void map_t_attributes(int choice);
void display_Ticket(void);


/*****************************************************************************
 * IsCorrectTicketNo
 ****************************************************************************/
// Checks length and character (only '0' to '9' are allowed)

 bool isCorrectTicketNo(char *n) {

     bool result = false;
     int x;
     int y = 0;

     // check for characters - only 0-9 are allowed
     for(int i = 0; i < 7; i++) {
         x = n[i] - 0;
         if(x > 47 && x < 58)
            y++;
         else
            break;
     }

     if(y == 7)
         result = true;
     return result;
 }

 /******************************************************************************
 * IsCorrectDateFormat
 *****************************************************************************/

bool isCorrectDateFormat(int m, int d, int y) {

    int ii = 0;        // incremented by 1 for each positive validation
    int febdays;    // used for number of days depending on year

    /* Checking month '''''''''''''''''''''''''''''''''''''''''''''''''''''' */
    if(m > 0 && m < 13)
        ii++;
    else
        printf("\nInvalid value for month!\n");

    /* Checking day depending on month '''''''''''''''''''''''''''''''''''''' */

    // Jan, Mar, May, Jul, Aug, Oct, Dec
    if(m==1|| m==3 || m==5 || m==7 || m==8 || m==10 || m==12) {
        if(d > 0 && d < 32)
            ii++;
        else
            printf("\nInvalid value for day!\n");
    }

    // Feb (validating leap years)

    if((isLeapYear(y)) == true)
        febdays = 29;
    else
        febdays = 28;

    if(m==2) {
        if(d > 0 && d < (febdays + 1))
            ii++;
        else
            printf("\nInvalid value for day!\n");
    }

    // Apr, Jun, Sep, Nov
    if(m==4|| m==6 || m==9 || m==11) {
        if(d > 0 && d < 31)
            ii++;
        else
            printf("\nInvalid value for day!\n");
    }

    /* Checking year ''''''''''''''''''''''''''''''''''''''''''''''''''''''' */
    if(y > 999 && y < 9999)                                // to ensure 4 digits
        ii++;
    else
        printf("\nInvalid value for year!\n");

    /* Evaluating complete result '''''''''''''''''''''''''''''''''''''''''' */

    if(ii == 3)
        return true;
    else
        return false;

}

/*****************************************************************************
 * IsLeapYear
 *****************************************************************************/
```

```

bool isLeapYear(const int iYear)
{
  // Each year that could be devided by 4 without rest is a leap year.
  // Exception: year could be devided by 100 without rest but not by 400.

  if ((iYear % 400) == 0)
    return true;
  else if ((iYear % 100) == 0)
    return false;
  else if ((iYear % 4) == 0)
    return true;

  return false;
}

/******************************************************************************
 * MAP TICKET ATTRIBUTES
 ******************************************************************************/

 /* Map some structure data first to some more human readable values for display on stdout.
    This is required since the input data for some ticket attributes differs from the output
    format. E.g. y => yes

    1: maps     current.T_Runtime    => display_runtime
    2: maps     current.T_D_Day        => display_weekday
    3: maps     current.T_G77        => display_G77
    4: maps     current.T_SU6        => display_SU6
    5: maps     current.T_GSP        => display_GSP

 */

 void map_t_attributes(int choice) {


    // range of allowed values
    char runtime_o1[2] = "1", runtime_o2[2] = "2", runtime_o3[2] = "3",
    runtime_o4[2] = "4", runtime_o5[2] = "5", runtime_o6[2] = "m", runtime_o7[2] = "p";
    char weekday_o1 [2] = "s", weekday_o2 [2] = "w", weekday_o3 [2] = "b";
    char T_G77_o1[2] = "y", T_G77_o2[2] = "n";
    char T_SU6_o1[2] = "y", T_SU6_o2[2] = "n";
    char T_GSP_o1[2] = "y", T_GSP_o2[2] = "n";

     switch(choice) {

         // Runtime
         case 1:

            if(strcmp(current.T_Runtime, runtime_o1) == 0) strcpy(display_runtime, "1 week");
            else if(strcmp(current.T_Runtime, runtime_o2) == 0) strcpy(display_runtime, "2 weeks");
            else if(strcmp(current.T_Runtime, runtime_o3) == 0) strcpy(display_runtime, "3 weeks");
            else if(strcmp(current.T_Runtime, runtime_o4) == 0) strcpy(display_runtime, "4 weeks");
            else if(strcmp(current.T_Runtime, runtime_o5) == 0) strcpy(display_runtime, "5 weeks");
            else if(strcmp(current.T_Runtime, runtime_o6) == 0) strcpy(display_runtime, "month");
            else if(strcmp(current.T_Runtime, runtime_o7) == 0) strcpy(display_runtime, "permanent");
            else strcpy(display_runtime, "");

         break;

         // Weekday
         case 2:
            if(strcmp(current.T_D_Day, weekday_o1) == 0) strcpy(display_weekday, "Sat");
            else if(strcmp(current.T_D_Day, weekday_o2) == 0) strcpy(display_weekday, "Wed");
            else if(strcmp(current.T_D_Day, weekday_o3) == 0) strcpy(display_weekday, "Wed + Sat");
            else strcpy(display_weekday, "");
         break;
```

```c

         // Game 77
         case 3:

            if(strcmp(current.T_G77, T_G77_o1) == 0) strcpy(display_G77, "yes");
            else if(strcmp(current.T_G77, T_G77_o2) == 0) strcpy(display_G77, "no");
            else strcpy(display_G77, "");

         break;

         // Super 6
         case 4:

            if(strcmp(current.T_SU6, T_SU6_o1) == 0) strcpy(display_SU6, "yes");
            else if(strcmp(current.T_SU6, T_SU6_o2) == 0) strcpy(display_SU6, "no");
            else strcpy(display_SU6, "");

         break;

         // Glueckspirale
         case 5:

            if(strcmp(current.T_GSP, T_GSP_o1) == 0) strcpy(display_GSP, "yes");
            else if(strcmp(current.T_GSP, T_GSP_o2) == 0) strcpy(display_GSP, "no");
            else strcpy(display_GSP, "");

         break;

         default: ;

     }
 }


 /*****************************************************************************
 * Display Ticket
 ****************************************************************************/
/* Displays ticket on terminal to confirm selection. This function can be used
for new tickets before writing ticket to file system or for existing tickets.
A loop for cofirming the selection is NOT part of the function itself. Note
that this function requires that the actual ticket data to display are already
expected to be in the current global ticket structure. */

void display_Ticket(void) {

    int i;

    /* Map some structure data first to some more human readable values for display on stdout */

    map_t_attributes(1);    // Runtime
    map_t_attributes(2);    // Weekday
    map_t_attributes(3);    // Game 77
    map_t_attributes(4);    // Super 6
    map_t_attributes(5);    // Glueckspirale

    // Display ticket ----------------------------------------

    printf("\n===============================================================================\n\n");
    printf("  Ticket No: %-11sPlayer: %-26s Active Rows: %-10d\n", current.T_No, current.T_Player, current.T_Max_Row);
    printf("  Date: %-16sTicket Runtime: %-19sWeekday: %-10s\n", current.T_Start, display_runtime, display_weekday);
    printf("  Game 77: %-13sSuper 6: %-26sGlueckspirale: %-10s\n", display_G77, display_SU6, display_GSP);
    printf("\n===============================================================================\n\n");

    for(i = 0; i < 12; i++) {

        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
    }

    printf("\n===============================================================================\n\n");

}

 /*******************************************************************************
 * IsCorrectLotteryRow
 ******************************************************************************
 checks for valid range and duplicates                                       */

bool isCorrectLotteryRow(int *LotteryNo) {

    int i, j; int v = 0; bool bDuplicate = false;

    // checks for duplicates
    for(i = 0; i < 6; i++) {
        for(j = 0; j < 6; j++) {
            if(i != j)
                if(LotteryNo[i] == LotteryNo[j])
                    bDuplicate = true;
        }
    }


    // checks for valid range
    for(i = 0; i < 6; i++) {
        if(LotteryNo[i] > 0 &&  LotteryNo[i] < 50)
            v++;
    }

    if((v == 6) && (bDuplicate == false))
        return true;

    return false;
}

 /*******************************************************************************
 * Convert To Digit
 ******************************************************************************
 converts single char in range of '1' to '9' to number.                        */

 /* returns int value if in range '0'..'9' else returns -1 if not a number */

int convertToDigit( char c )
{
    if ( c < '0' || c > '9' ) return -1;
    return c - '0';
}

 /*******************************************************************************
 * Get Lottery Win Class
 ******************************************************************************
 Function gets number of lottery matches per row and result from check of super
 super number per ticket. The fuction returns the lottery win class.           */

 char *getWinClass(int matches, bool super ) {

     if(super == false) {

         switch(matches) {

             case 0: return "no win"; break;
             case 1: return "no win"; break;
             case 2: return "no win"; break;
             case 3: return "class VIII"; break;
```

```c
                case 4: return "class VI"; break;
                case 5: return "class IV"; break;
                case 6: return "class II"; break;
                default: return "no win";
            }

        } else {

            switch(matches) {

                case 0: return "no win"; break;
                case 1: return "no win"; break;
                case 2: return "class IX"; break;
                case 3: return "class VII"; break;
                case 4: return "class V"; break;
                case 5: return "class III"; break;
                case 6: return "class I"; break;
                default: return "no win";
            }
        }
}


 /******************************************************************************
 * IsValidDrawingDate
 *****************************************************************************
 Check for ticket validity against drawing date. Gets Drawing Date and returns
 true or false by checking Ticket Start Date, Ticket Runtime and Ticket day of
 week - all stored in the global ticket structure. In order to accomplish this,
 the functions first needs to calculate the ticket end date. */

bool isValidDrawingDate(int dd_month, int dd_day, int dd_year) {

    bool result;                    // true if w_day_OK AND period_OK true. Otherwise false.
    bool w_day_OK;                   // true if drawing day of the week matches with ticket. Otherwise false.
    bool period_OK;                  // true if with ticket validity period. Otherwise false.
    struct tm add;                  // date type used for a_ctual d_rawing d_ate
    struct tm tsd;                  // date type used for t_icket s_tart d_ate
    struct tm ted;                  // date type used for t_icket e_nd d_ate
    int ts_day, ts_month, ts_year;  // representing year, month, day from ticket start date as integer values
    char *str_day;                  // Token of Ticket start date for day
    char *str_month;                // Token of Ticket start date for month
    char *str_year;                 // Token of Ticket start date for year
    char str_runtime;               // runtime range: 1,2,3,4,5,m_onth, p_ermanent
    /* char str_d_day;              drawing day range: s_saturday, w_ednesday or b_oth*/
    double diff_seconds1;           // time difference in seconds
    double diff_seconds2;           // time difference in seconds
    int dw_day;                     // drawing weekday (0,1,2,3,4,5,6)
    char tw_day;                    // ticket week day (s, w or b)


    result = false;


    // build actual drawing date structure ------------------------------------

    add.tm_year = dd_year - 1900;
    add.tm_mon  = dd_month - 1;
    add.tm_mday = dd_day;

    add.tm_hour = 0;
    add.tm_min  = 0;
    add.tm_sec  = 1;
    add.tm_isdst = -1;        // Change for Summer Time !?

    if (mktime(&add) == -1 )
```

```
        add.tm_wday = 7;

    // build ticket start date structure --------------------------------------

    str_day = strtok(current.T_Start, ".");
    str_month = strtok(NULL, ".");
    str_year = strtok(NULL, ".");

    ts_day = atoi(str_day);
    ts_month = atoi(str_month);
    ts_year = atoi(str_year);

    tsd.tm_year = ts_year - 1900;
    tsd.tm_mon  = ts_month - 1;
    tsd.tm_mday = ts_day;

    tsd.tm_hour = 0;
    tsd.tm_min  = 0;
    tsd.tm_sec  = 1;
    tsd.tm_isdst = -1;       // Change for Summer Time !?

    if (mktime(&tsd) == -1 )
        tsd.tm_wday = 7;

    // build ticket end date structure ----------------------------------------

    str_runtime = current.T_Runtime[0];
    //str_d_day = current.T_D_Day[0];

    ted.tm_year = ts_year - 1900;
    ted.tm_mon  = ts_month - 1;
    ted.tm_mday = ts_day;

    ted.tm_hour = 0;
    ted.tm_min  = 0;
    ted.tm_sec  = 1;
    ted.tm_isdst = -1;       // Change for Summer Time !?

    switch (str_runtime) {

        case '1': ted.tm_mday +=7; mktime(&ted); break;     // 1 week = 7 days
        case '2': ted.tm_mday +=14; mktime(&ted); break;    // 2 weeks = 14 days
        case '3': ted.tm_mday +=21; mktime(&ted); break;    // 3 weeks = 21 days
        case '4': ted.tm_mday +=28; mktime(&ted); break;    // 4 weeks = 28 days
        case '5': ted.tm_mday +=35; mktime(&ted); break;    // 5 weeks = 35 days
        case 'm': ted.tm_mon +=1; mktime(&ted); break;      // 1 month = same date next month
        case 'p': ted.tm_year +=25; mktime(&ted); break;    // permanent = 25 years
        default: break;
    }


    // Check validity ---------------------------------------------------------
    // Condition 1: weekday 0=Sun, 1=Mon, 2=Tue, 3=Wed, 4=Thu, 5=Fri, 6=Sat
    // Condition 2: Ticket end date is greater or equals actual drawing date
    // Condition 3: Actual drawing date is grater or equals ticket start date
    // Ticket has been validated against drawing date once all conditions above are evaluated true

    diff_seconds1 = difftime(mktime(&add), mktime(&tsd));       // difference between actual drawing date and ticket start d
    diff_seconds2 = difftime(mktime(&ted), mktime(&add));       // difference between ticket end date and actual drawing dat

    dw_day = add.tm_wday;                                       // day of the week for actual drawing date from console inpu
    tw_day = current.T_D_Day[0];                                // day of the week read from ticket (w, s or b)

    // Check for correct weekday (condition 1)

    w_day_OK = false;
```

```c
    period_OK = false;


    switch (tw_day) {

        case 's':

            if(dw_day == 6) w_day_OK = true;
            else printf("\nDrawing date is not a Saturday. Please enter correct drawing date.\n");
            break;

        case 'w':

            if(dw_day == 3) w_day_OK = true;
            else printf("\nDrawing date is not a Wednesday. Please enter correct drawing date.\n");
            break;

        case 'b':

            if(dw_day == 3 || dw_day == 6) w_day_OK = true;
            else printf("\nDrawing date is neither Wednesday nor Saturday. Please enter correct drawing date.\n");
            break;

        default:

            printf("\nDay of Week from ticket not available. \n");
            break;
    }



    // Check for correct validity period (condition 2 & 3)

    if((diff_seconds1 >= 0) && (diff_seconds2 >= 0)) period_OK = true;
    else printf("\nActual drawing date not in rage of valid ticket period. Please enter correct drawing date.\n");

    // Total result for check of validity

    if(w_day_OK == true && period_OK == true) result = true;
    else result = 0;

    /* Debug
    printf("DEBUG add: %s\n", asctime(&add));
    printf("DEBUG tsd: %s\n", asctime(&tsd));
    printf("DEBUG ted: %s\n", asctime(&ted));
    printf("DEBUG diff_seconds1: %f\n", diff_seconds1);
    printf("DEBUG diff_seconds2: %f\n", diff_seconds2);
    printf("DEBUG dw_day: %d\n", dw_day);
    printf("DEBUG tw_day: %c\n", tw_day); */

    return result;

}
```

t_delete.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

```c
 /*t_evaluate.c | RLotto | gcc | v0.8.354.1715
 * Console program for storing and evaluating lottery ticket resultt.
 * ---------------------------------------------------------------------------
 *
 * Objective:    Evaluate selected ticket based on corresponding draw results
 *
 * Author:          Reinhard Rozumek
 * Email:          reinhard@rozumek.de
 * Created:      10/08/17
 * Last mod:     02/11/18
 *
 * ---------------------------------------------------------------------------
 * 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 <time.h>
#include "rlotto.h"
#include "version.h"


/* GLOABL VARIABLES ***********************************************************/

struct tm dd;                                      // date type used for d_rawing d_ate

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

int enterInput(void);                                           // enter drawing results
int checkLotto(void);                                           // evaluate lottery result
int checkGame77(void);                                           // evaluate Game 77 result
int checkSuper6(void);                                           // evaluate Super 6 result
int checkGSP(void);                                               // evaluate Glueckspirale result
int isCorrectDateFormat(int m, int d, int y);                   // validating date format
bool isValidDrawingDate(int dd_month, int dd_day, int dd_year); // validating date range
bool isCorrectLotteryRow(int *LotteryNo);                       // checks valid range & duplicates
int convertToDigit( char c );                                   // converts single char in range of '1' to '9' to number.
char *getWinClass(int matches, bool bonus_super );               // returns win class based on lottery matches

 /*    ***************************************************************************
    EVALUATE TICKET
    ***************************************************************************/

/*  Enter actuals drawings results and evaluate against selected ticket. This
    function create the output/result file and calls all sub functions for ticket
    evaluation. */


    // FIXME: (camelo#1#01/20/18): Adjust for new rules - remove Bonus number adjust win classed by Superzahl
    // TODO: (camelo#2#01/03/18): Implement condition to evaluate only enabled options (e.g. G77)

int evaluateTicket(void) {

    int sConfirm;                               // yes or no to confirm user input
    char sPath[45];                            // Full pathname for result file
    char sPrefix[14] = "Lotto-Result-";        // File prefix
    char sPostfix[5] = ".txt";                 // File extension
    char sDrwDate[25];                         // Drawing Date formatted as part of filename

    printf("\n\nEnter actual drawing result.\n");

    // Call function to enter drawing results
    enterInput();

    // Final confirmation for starting evaluation
    printf("\nEvaluate results now? [y/n]: ");

    do {
        sConfirm = tolower(getchar());
    } while(sConfirm != 'y' && sConfirm != 'n');

    if(sConfirm == 'y') {

        // Create Filename
        strftime(sDrwDate, N, "%Y-%m-%d", &dd); strcpy(sPath, ResultFolder); strcat(sPath, sPrefix);
         strcat(sPath, current.T_No); strcat(sPath, "-"); strcat(sPath, sDrwDate); strcat(sPath, sPostfix);

        // Open result file for output
        pFile = fopen(sPath, "w");
        if(pFile == NULL) {
            printf("\nResult folder missing. Try to create now...\n");
            system("mkdir results");
            pFile = fopen(sPath, "w");
            if(pFile == NULL) {
                printf("Error opening %s for writing. Program terminated.", sPath);
                abort();
            } else {
                printf("Folder \"%s\" has been created.\n", ResultFolder);
            }
        }

        // Output to result file first part (header information)
        fprintf(pFile, "%s v%ld.%ld.%ld.%ld\n", THISPROG,MAJOR,MINOR,BUILD,REVISION);
        fprintf(pFile, "Evaluating lottery results\n");
        fprintf(pFile, "Lottery Ticket No: %s\n", current.T_No);

        fprintf(pFile, "\nPlayers: %s\n", current.T_Player);

        strftime(sDrwDate, N, "%A, %d-%b-%Y", &dd);

        fprintf(pFile, "\nDrawing Date: %s\n", sDrwDate);
        fprintf(pFile, "Lottery numbers: %i %i %i %i %i %i\n",ALN[0],ALN[1],ALN[2],ALN[3],ALN[4],ALN[5]);
        fprintf(pFile, "Super number: %i\n", ASN);
        fprintf(pFile, "Game 77: %s\n", cG77);
        fprintf(pFile, "Super 6: %s\n", cSU6);

        fprintf(pFile, "\nLottery Matches on %s\n\n", sDrwDate);

        // Evaluate results
        checkLotto(); checkGame77(); checkSuper6();
        printf("\nWithout any warranty.\n");

        // Final output for result file
        fprintf(pFile, "\nWithout any warranty.\n");

         // Close result file
        fclose(pFile);
        printf("\nResults written to %s.\n", sPath);

    }

    return 0;
}


```

```
/*******************************************************************************
 * Enter Input
 *******************************************************************************/

int enterInput() {

     // Actual Play Date (drawing date) ''''''''''''''''''''''''''''''''''''''''

    int year, month, day;            // year, month, day as enterd by user
    bool is_ok = false;             //
    bool date_format_ok = false;    // correctness of drawing date format
    bool date_range_ok = false;     // validity of drawing date related to ticket
    char sPlayDate[40];                // Actual Drawing Date
    int i;                            // Actual Lottery Number Index
    bool first_input = true;        // indicates first attempt for input
    char sDrwDate[N];                // Drawing date formated


    while(date_format_ok == false || date_range_ok == false)
    {

        if(first_input == true)
            printf("\nEnter drawing date (mm/dd/yyyy): ");
        else
            printf("Invalid input for drawing date! Please correct: ");
        scanf("%d/%d/%d", &month, &day, &year);
        fflush(stdin);

        dd.tm_year = year - 1900;
        dd.tm_mon  = month - 1;
        dd.tm_mday = day;

        dd.tm_hour = 0;
        dd.tm_min  = 0;
        dd.tm_sec  = 1;
        dd.tm_isdst = -1;       // Change for Summer Time !?

        if (mktime(&dd) == -1 )
          dd.tm_wday = 7;

        strftime(sPlayDate, 40, "%A, %d-%b-%Y", &dd);

        date_format_ok = isCorrectDateFormat(month, day, year);
        date_range_ok = isValidDrawingDate(month, day, year);



        first_input = false;

    }


    first_input = true;    // reset to true for next evaluation

     // Actual Lottery Numbers ''''''''''''''''''''''''''''''''''''''''''''''''

     do {
        // Initialize ALN
        for(i = 0; i < 6; ++i) {
            ALN[i] = 0;
        }


         if(first_input == true)
             printf("Enter actual lottory numbers seperated by commas: ");
         else
```

```
                printf("Invalid input! /Lottery Numbers) Please correct: ");
            scanf("%i,%i,%i,%i,%i,%i", &ALN[0], &ALN[1], &ALN[2], &ALN[3], &ALN[4], &ALN[5]);
            fflush(stdin);
            is_ok = (isCorrectLotteryRow(ALN));
            first_input = false;

        } while(is_ok == false);

        first_input = true;    // reset to true for next evaluation

        // Actual Lottery Super Number '''''''''''''''''''''''''''''''''''''''''''

        do {
            if(first_input == true)
                printf("Enter actual super number: ");
            else
                printf("Invalid input! Please correct: ");
            scanf("%i", &ASN);
            fflush(stdin);
            first_input = false;

        } while(ASN < 0 || ASN > 9);

        first_input = true;    // reset to true for next evaluation

        // Actual Game 77 '''''''''''''''''''''''''''''''''''''''''''''''''''''''

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

            do {
                if(first_input == true)
                    printf("Enter actual Game 77 number: ");
                else
                    printf("Invalid input! Please correct: ");
                scanf("%7[0123456789]", cG77);
                fflush(stdin);
                first_input = false;

            } while((strlen(cG77)) < 7);

            first_input = true;    // reset to true for next evaluation
        }

        // Actual Super 6 '''''''''''''''''''''''''''''''''''''''''''''''''''''''

       if(current.T_SU6[0] == 'y'){

           do {
                if(first_input == true)
                    printf("Enter actual Super 6 number: ");
                else
                    printf("Invalid input! Please correct: ");
            scanf("%6[0123456789]", cSU6);
            fflush(stdin);
            first_input = false;

            } while((strlen(cSU6)) < 6);

            first_input = true;    // reset to true for next evaluation
       }

        // Actual Glueckspirale '''''''''''''''''''''''''''''''''''''''''''''''''




```

```
     // Console Output

    strftime(sDrwDate, N, "%A, %d-%b-%Y", &dd);

    printf("\nCheck your input:\n");
    printf("\nDrawing Date: %s\n", sDrwDate);
     printf("Actual lottery numbers: %i %i %i %i %i %i\n",ALN[0],ALN[1],ALN[2],ALN[3],ALN[4],ALN[5]);
     printf("Actual super number: %i\n", ASN);
     if(current.T_G77[0] == 'y')
        printf("Actual Game 77: %s\n", cG77);
    else
        printf("Actual Game 77: N/A\n");
    if(current.T_SU6[0] == 'y')
         printf("Actual Super 6: %s\n", cSU6);
    else
        printf("Actual Super 6: N/A\n");
     return 0;

}

/*******************************************************************************
 * check Lotto
 *******************************************************************************

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



 int checkLotto() {

     int RowNo, i, j;                             // row number, indizees
     int MPR[NOLR];                                 // matches per lottery row
     int WinRows = 0;                             // number of lottery rows with win
     bool CSN;                                    // indicates correct super number matching last digit of the ticket number
     char sDrwDate[N];                             // Drawing date formated
     int iNOLR = current.T_Max_Row;               // Number of lottery rows active on the ticket
     char WinMsg[12];                             // literal to show win class for ouput

     // Initialize MPR and CSN

     for(i = 0; i < NOLR; i++) {
        MPR[i] = 0;
     }

    CSN = false;

// check for matches with actual lottery numbers ''''''''''''''''''''''''''''''

    for(RowNo = 0;RowNo < NOLR; RowNo++)
        for(i = 0; i < 6; i++)
            for(j = 0; j < 6; j++)
                if(current.T_Row[RowNo][i] == ALN[j])
                    MPR[RowNo]++ ;

    // check for correct Super Number ''''''''''''''''''''''''''''''''''''''''

    if(ASN == convertToDigit(current.T_No[6]))
                    CSN = true;

    // Generate output ''''''''''''''''''''''''''''''''''''''''''''''''''''''''

    // Console output + result file output for lottery results

    strftime(sDrwDate, N, "%A, %d-%b-%Y", &dd);
    printf("\nLottery Matches on %s\n\n", sDrwDate);
```

```c

    // Loop through lottery rows to generate console and file output for matches and win(s)

    for(RowNo = 0; RowNo < NOLR; RowNo++) {

        strcpy(WinMsg, getWinClass(MPR[RowNo], CSN));

        if(strcmp(WinMsg, "no win") != 0)
            WinRows++; //counting number of rows with win

        switch(MPR[RowNo]) {

            case 0:
                printf("Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / no match)\n",RowNo + 1,current.T_Row[RowNo][0], current.
                fprintf(pFile, "Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / no match)\n",RowNo + 1,current.T_Row[RowNo][0],
            break;

            case 1:
                printf("Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i match)\n",RowNo + 1,current.T_Row[RowNo][0], current.
                fprintf(pFile, "Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i match)\n",RowNo + 1,current.T_Row[RowNo][0],
            break;
            default:
                if(CSN == true) {
                    printf("Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i matches + correct super number)\n",RowNo + 1,curr
                    fprintf(pFile, "Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i matches + correct super number)\n",RowNo
                } else {
                    printf("Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i Matches)\n",RowNo + 1,current.T_Row[RowNo][0], cu
                    fprintf(pFile, "Row No %2i: %2i %2i %2i %2i %2i %2i\t(%s / %i Matches)\n",RowNo + 1,current.T_Row[RowNo
                }
            break;
        }
    }


    switch(WinRows) {

        case 0: {
            if(iNOLR == 1) {
                printf("\nThere is no win for any of %i row played in total.\n",iNOLR);
                fprintf(pFile, "\nThere is no win for any of %i row played in total.\n",iNOLR);
            } else {
                printf("\nThere is no win for any of %i rows played in total.\n",iNOLR);
                fprintf(pFile, "\nThere is no win for any of %i rows played in total.\n",iNOLR);
            }
            break;
        }
        case 1: {
            if(iNOLR == 1) {
                printf("\nThere is %i row with win of %i row played in total.\n",WinRows,iNOLR);
                fprintf(pFile, "\nThere is %i row with win of %i row played in total.\n",WinRows,iNOLR);
            } else {
                printf("\nThere is %i row with win of %i rows played in total.\n",WinRows,iNOLR);
                fprintf(pFile, "\nThere is %i row with win of %i rows played in total.\n",WinRows,iNOLR);
            }
            break;
        }
        default: {
            printf("\nThere are %i rows with wins of %i row(s) played in total.\n",WinRows,iNOLR);
        }
    }

     return 0;
}

/******************************************************************************
 * check Game 77
 ******************************************************************************

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

 int checkGame77() {

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


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

        for(ii = 6; ii >= 0; ii--) {
            if(current.T_No[ii] == cG77[ii])
                MatchG77++;
            else
            break;
        }

        if(MatchG77 > 0) {
            switch(MatchG77) {

                case 1: strcpy(WinClassG77, "VII"); break;
                case 2: strcpy(WinClassG77, "VI"); break;
                case 3: strcpy(WinClassG77, "V"); break;
                case 4: strcpy(WinClassG77, "IV"); break;
                case 5: strcpy(WinClassG77, "III"); break;
                case 6: strcpy(WinClassG77, "II"); break;
                case 7: strcpy(WinClassG77, "I"); break;
                default: strcpy(WinClassG77, "---");
            }

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

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

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

        }
    } else {

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

     return 0;
}

/******************************************************************************
 * check Super 6
 ******************************************************************************

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

 int checkSuper6() {


     int ii;                                           // index of ticket number array
     int MatchSU6 = 0;                               // matches Super 6
     char WinClassSU6[4];                           // Game 77 win class

    if(current.T_SU6[0] == 'y'){

        for(ii = 6; ii >= 0; ii--) {
        if(current.T_No[ii] == cSU6[ii -1])
            MatchSU6++;
        else
            break;
    }

        if(MatchSU6 > 0) {
            switch(MatchSU6) {

                case 1: strcpy(WinClassSU6, "VI"); break;
                case 2: strcpy(WinClassSU6, "V"); break;
                case 3: strcpy(WinClassSU6, "IV"); break;
                case 4: strcpy(WinClassSU6, "III"); break;
                case 5: strcpy(WinClassSU6, "II"); break;
                case 6: strcpy(WinClassSU6, "I"); break;
                default: strcpy(WinClassSU6, "---");
            }


            if(MatchSU6 == 1) {
                printf("You have won Super 6 according winning class %s (%i digit matching).\n",WinClassSU6,MatchSU6);
                fprintf(pFile, "You have won Super 6 according winning class %s (%i digit matching).\n",WinClassSU6,MatchSU
            }

            if(MatchSU6 > 1) {
                printf("You have won Super 6 according winning class %s (%i digits matching).\n",WinClassSU6,MatchSU6);
                fprintf(pFile, "You have won Super 6 according winning class %s (%i digits matching).\n",WinClassSU6,MatchS

            }

        } else {
            printf("There is no win for Super 6.\n");
            fprintf(pFile, "There is no win for Super 6.\n");

        }
    } else {

        printf("Super 6 is not appicable for this ticket.\n");
        fprintf(pFile, "Super 6 is not appicable for this ticket.\n");

    }

     return 0;
}

/*****************************************************************************
 * check GLUCKSPIRALE
 *****************************************************************************

 Evaluates ticket against actual result for 'Gluecksspirale'. */

 // TODO (camelo#1#01/03/18): Implement evaluation for Gluecksspirale

 int checkGSP() {


     return 0;
}
```

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
            {
                i++;
            }
        }

        closedir(d);
        i_max = i;
    }



    // Second run to read ticket file names in array for all files matching file extension
    d = opendir(TicketFolder);

    if (d)
    {
        i = 0;                                                  // Reset counter i
        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
            {
                i++;
                strcpy(fn_array[i-1],fn);
                printf("%3i --> Ticket Number: %.*s File Name: %s\n", i, 7, fn + 0, fn_array[i-1]);
            }
        }
        closedir(d);
    }


    // i equals null --> no ticket found
    if(i == 0)
    {
         // no ticket found
         printf("\nNo ticket found in selected directory.\n");
    }
    else
    {
        printf("\n%i ticket(s) found in selected directory.\n", i_max);
        do {
            printf("\nEnter number between %i and %i: ",i_min, i_max);
            scanf("%d", &i_input);
            if(i_input < 1 || i_input > i_max)
            if(i_input < 1 || i_input > i_max)
            {
                printf("\n \"%d\" is not between %i and %i !\n",i_input, i_min, i_max);
                printf("\n \"%d\" is not between %i and %i !\n",i_input, i_min, i_max);
            }
        }while(i_input < 1 || i_input > i_max);

        strncpy(ticket_no, fn_array[i_input - 1] + 0,7);
        ticket_no[7] = '\0';
        printf("\nTicket No %s has been selected.\n", ticket_no);
    }



    // Call function for reading selected ticket

    read_Ticket(ticket_no);

    return 0;
}

/*    READ TICKET ***************************************************************
```

```
    Reads selected ticket into ticket structure
    ****************************************************************************/

    void read_Ticket(char *ticket_no) {

        FILE *fp;                          // file pointer
        char line[MAX_LINE_LENGTH];          // array for line string
        int lnr;                            // line number of input file
        char *description;                  // left token of separated input line contains line description
        char *value;                      // right token of separated input line contains line value
        char *token1, *token2, *token3;     // token 1-3 for lottery numbers
        char *token4, *token5, *token6;     // token 4-6 for lottery numbers
        int sConfirm;                      // yes or no to confirm ticket for further evaluation


        // construct filename
        char t_filename[45];
        strcpy(t_filename, TicketFolder);
        strcat(t_filename, ticket_no);
        strcat(t_filename, T_EXT);

        // Open file for reading ----------------------------------

        lnr = 0;
        fp = fopen(t_filename, "r");

        if(fp == NULL) {

            printf("Error opening file!\n");

        } else {

            while(fgets(line, MAX_LINE_LENGTH, fp) != NULL){

                strtok(line, "\n");            // Remove trailing new line character

                lnr++;

                // Parsing input and copy into current ticket structure ----------------------------------

                // separate input line

                description = strtok(line, ":");
                value = strtok(0, ":");

                // reading values into ticket structure

                switch(lnr){

                    case 1:
                        strcpy(current.T_No, value); break;
                    case 2:
                        strcpy(current.T_Player, value); break;
                    case 3:
                        strcpy(current.T_Start, value); break;
                    case 4:
                        strcpy(current.T_Runtime, value); break;
                    case 5:
                        strcpy(current.T_D_Day, value); break;
                    case 6:
                        strcpy(current.T_G77, value); break;
                    case 7:
                        strcpy(current.T_SU6, value); break;
                    case 8:
                        strcpy(current.T_GSP, value); break;
                    case 9:
                        current.T_Max_Row = atoi(value); break;
                    case 10:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 11:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 12:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 13:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 14:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 15:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 16:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 17:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 18:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 19:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 20:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    case 21:

                        token1 = strtok(value, ","); current.T_Row[lnr -10][0] = atoi(token1);
                        token2 = strtok(0, ","); current.T_Row[lnr -10][1] = atoi(token2);
                        token3 = strtok(0, ","); current.T_Row[lnr -10][2] = atoi(token3);
                        token4 = strtok(0, ","); current.T_Row[lnr -10][3] = atoi(token4);
                        token5 = strtok(0, ","); current.T_Row[lnr -10][4] = atoi(token5);
                        token6 = strtok(0, ","); current.T_Row[lnr -10][5] = atoi(token6);
                        break;

                    default:
                    printf("\n Error parsing ticket file.\n");

                }

        }


        fclose(fp);

    }
```

```

    // Display selected Ticket ----------------------------------

    display_Ticket();

    // confirm for evaluation ----------------------------------

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

    do {
        sConfirm = tolower(mygetc());
        if(sConfirm != 'y' && sConfirm != 'n')
            printf("\nEnter \"y\" or \"n\"\n");
        if(sConfirm == 'y')
            evaluateTicket();
        if(sConfirm == 'n') {
            printf("\nReturning to main menu.\n");
            t_initialize();
            // exit statement removed due to compiler warning (
        }

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

}
```

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.
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
""")
parser.add_argument('--header',  help='quoted, space separated list of *.h files',  required=True)
parser.add_argument('--csource', help='quoted, space separated list of *.c files',  required=True)
parser.add_argument('--python',  help='quoted, space separated list of *.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}
{\\Huge{\\textbf{RLOTTO Code Manual}}}
\\newline
"""

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

print(LATEX1)
print("\\texttt{" + 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:
        t = f.replace("_","\\_")
        print("\\subsection*{\\texttt{" + t + "}}")
        print("\\addcontentsline{toc}{subsection}{\\texttt{" + t + "}}")
        print("\\inputminted{C}{../" + f + "}\\newpage")

if args.python:
    t = args.python.replace("_","\\_")
    print("\\subsection*{\\texttt{" + t + "}}")
    print("\\addcontentsline{toc}{subsection}{\\texttt{" + t + "}}")
    print("\\inputminted{python}{" + args.python + "}\\newpage")


# example ticket
print("\\section{Examples}")
print("\\subsection*{Ticket Input}")
print("\\addcontentsline{toc}{subsection}{Ticket Input}")
print("\\inputminted{text}{{../1234567.tck}}\\newpage")

# exmaple results
print("\\subsection*{Output Result}")
print("\\addcontentsline{toc}{subsection}{Output Result}")
print("\\inputminted{text}{{../results/Lotto-Result-2017-11-12.txt}}\\newpage")

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.
```