```c
/*  GENERAL INFORMATION

 Project created by Sara Marfella IST188316 on May 16, 2017

* Implemented:
* 1. Load the data structures required to represent
*    the stations and the trips.
* 2. Read parameters of main and 2 files required
*    to make the program work.
* 3. Build 2 menus and data input.
* 4. Build the lists of trips and stations,
*    inserting items in an ordered way.
* 5. Implementation of trips list mode.
* 6. Implementation of data filters.
* 7. Implementation of stations list mode.
* 8. Implementation of routes list mode.
*
* Not Implemented:
* 9. Implementation of graphic mode.
* 10. Implementation of statistical analysis.
* 11. Improvements of application graphics mode,
*     with a real map.
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>
#include <string.h>
#include "dataManager.h"
#include "print.h"

// EVIL GLOBAL VARIABLES
int filter_hour_start = -1;
int filter_hour_end = -1;

char trips_file[MAX_SIZE];
char stations_file[MAX_SIZE];

// DECLARATION OF FUNCTIONS

// command line interface
void mainMenu(Trip*, Station*);
void listOfStationsMenu(Trip *, Station *);
void selectDataMenu();
void selectStationMenu(Station*, Trip*, int);
void statsMenu();
void clearInputBuffer();

/* Main Function
 * \param mode and file (trips file  and stations file)
 */
int main (int argc, char *argv[]){

    // check for correct input
    if (argc != 4) {
        printf("Error - required arguments missing.\n");
        printf("Please start with -t trips-filename stations-filename\n");
        return 0;
    }
    if (strcmp(argv[1], "-t") != 0) {
        printf("Sorry - only text mode is available,
        printf("Please start with -t\n");
        return 0;
    }

```

```c
    strcpy(trips_file, argv[2]);
    strcpy(stations_file, argv[3]);

    Trip * allTrips = readTripsData(trips_file);
    Station * allStations = readStationData(stations_file);

    mainMenu(allTrips, allStations);

    return 0;
}

/* Command Line Interface */

/* mainMenu: prints the main menu and handles the main menu options
 * \param tripList      the header of the trip list
 *                      (can be the filtered list)
 * \param stationsList  the header of the stations list
 */
void mainMenu(Trip * tripsList, Station * stationsList){
    int command;
    printf("\n * M A I N   M E N U * \n\n");
    printf("  [ 1 ]   Select the data\n\n");
    printf("  [ 2 ]   Print List of trips\n\n");
    printf("  [ 3 ]   Print List of stations\n\n");
    printf("  [ 4 ]   Print List of routes\n\n");
    printf("  [ 5 ]   Print List of statistics\n\n");
    printf("  [ 6 ]   Quit\n\n");
    scanf("%d", &command);
    clearInputBuffer();
    switch (command) {
        case 1:
            selectDataMenu(tripsList, stationsList);
            break;
        case 2:
            command = -1;
            while ((command < 0) || (command > 32000)) {
                printf("How many trips do you want to print?
                printf("0..32000, enter 0 for all)\n");
                scanf("%d", &command);
                clearInputBuffer();
            }
            printTripsList(tripsList, command);
            mainMenu(tripsList, stationsList);
            break;
        case 3:
            listOfStationsMenu(tripsList, stationsList);
            break;
        case 4:
            selectStationMenu(stationsList, tripsList, 0);
            break;
        case 5:
            statsMenu();
            mainMenu(tripsList, stationsList);
            break;
        case 6:
            exit(EXIT_SUCCESS);
            break;
        default:
            printf("Error: invalid command...\n");
            mainMenu(tripsList, stationsList);
            break;
    }
}

/* List of Stations Menu:   creates and prints the list of
 *                          stations with max/min/avg
```

```
 * \param tripList          the header of the trip list
 *                          (usually the filtered list)
 * \param stationsList      the header of the stations list
 */
void listOfStationsMenu(Trip * tripsList, Station * stationsList){
    Station * filteredStations = countBikes(tripsList,
        stationsList, filter_hour_start, filter_hour_end);

    // option to ask the user: limit (0 for all),
    //should print stations with no trips (YES, NO)
    printStationsList(filteredStations, 0, YES);
    mainMenu(tripsList, stationsList);
}

/* SelectDataMenu: prints and manages the menu to set
*                  search criteria
 * \param filteredTrips     the header of the trip list
                            (can be the filtered list)
 * \param allStations       the header of all stations list
 */
void selectDataMenu(Trip * filteredTrips, Station * allStations){

    int duration = -1, command;

    printf("\n * Select the mode of your search * \n\n");
    printf("  [ 1 ]   Period of time (hour start, hour end)\n\n");
    printf("  [ 2 ]   Day of week\n\n");
    printf("  [ 3 ]   Max duration of trip (in seconds)\n\n");
    printf("  [ 4 ]   New Search (reset list)\n\n");
    printf("  [ 5 ]   Return to Main Menu\n\n");
    scanf("%d", &command);
    clearInputBuffer();
    switch (command) {
        case 1:
            command = -1;

            while ((command < 0) || (command > 23)) {
                printf("Insert start time of trip (hour 0..23):\n");
                scanf("%d", &command);
                clearInputBuffer();
            }
            // temporary place to store start time cannot
            //overwrite the global now
            // because we use it to understand if user has
            //used the filter before, later in the code
            int start = command;

            command = -1;
            while ((command < 0) || (command > 23)) {
                printf("Insert end time of trip (hour 0..23):\n");
                scanf("%d", &command);
                clearInputBuffer();
            }
            filter_hour_end = command;

            // if user had filtered the list before,
            // reset the list to avoid problems
            if (filter_hour_start != -1) {
                // now we can save the start time
                filter_hour_start = start;
                Trip * allTrips = readTripsData(trips_file);
                printf("Note: a time filter was already applied.\n
                printf("The trip list was reset to use the new param.\n");
                filteredTrips = selectTripsByTime(allTrips,
                                filter_hour_start, filter_hour_end);
            } else {
```

```
199                 filter_hour_start = start;
200                 filteredTrips = selectTripsByTime(filteredTrips,
201                                 filter_hour_start, filter_hour_end);
202             }
203             mainMenu(filteredTrips, allStations);
204             break;
205         case 2:
206             command = 0;
207             while ((command < 1) || (command > 7)) {
208                 printf("\nPlease insert the day of trip:\n\n");
209                 printf("  [ 1 ]   Monday\n");
210                 printf("  [ 2 ]   Tuesday\n");
211                 printf("  [ 3 ]   Wednesday\n");
212                 printf("  [ 4 ]   Thursday\n");
213                 printf("  [ 5 ]   Friday\n");
214                 printf("  [ 6 ]   Saturday\n");
215                 printf("  [ 7 ]   Sunday\n");
216                 scanf("%d", &command);
217                 clearInputBuffer();
218             }
219             filteredTrips = selectTripsByDay(filteredTrips, command);
220             mainMenu(filteredTrips, allStations);
221             break;
222         case 3:
223             while ((duration < 0) || (duration > 32767)) {
224                 printf("Insert the max duration of trip (in seconds):\n");
225                 scanf("%d", &duration);
226                 clearInputBuffer();
227             }
228             filteredTrips = selectTripsByDuration(filteredTrips,duration);
229             mainMenu(filteredTrips, allStations);
230             break;
231         case 4:
232             filter_hour_end = -1;
233             filter_hour_start = -1;
234             Trip * allTrips = readTripsData(trips_file);
235             mainMenu(allTrips, allStations);
236             break;
237         case 5:
238             if (filteredTrips != NULL)
239                 mainMenu(filteredTrips, allStations);
240             else {
241                 Trip * allTrips = readTripsData(trips_file);
242                 mainMenu(allTrips, allStations);
243             }
244             break;
245         default:
246             printf("Error: invalid command...\n");
247             if (filteredTrips != NULL)
248                 selectDataMenu(filteredTrips, allStations);
249             else {
250                 Trip * allTrips = readTripsData(trips_file);
251                 selectDataMenu(allTrips, allStations);
252             }
253             break;
254     }
255 }
256
257 /* selectStationMenu:
258  * \param allStations      the header of all stations list
259  * \param filteredTrips    the header of the trips list (can be filtered)
260  * \param id               the ID of the station
261  */
262 void selectStationMenu(Station * allStations,Trip * filteredTrips,int id){
263     int command = -1;
264     char stationName[7] = "";
```

```c
265
266     // validate id of station
267     while (strcmp(stationName, "") == 0) {
268         printf("Insert the id of the station:\n");
269         scanf("%d", &id);
270         clearInputBuffer();
271         strcpy(stationName, getStationNameById(id, allStations));
272     }
273     filteredTrips = selectTripsByIdStation(filteredTrips, id);
274
275     command = -1;
276     while ((command <0) || (command > 32767)) {
277         printf("How many routes do yo want to print? (0 for all)\n");
278         scanf("%d", &command);
279         clearInputBuffer();
280     }
281     printRoutesList(filteredTrips, allStations, id, command);
282     Trip * allTrips = readTripsData(trips_file);
283     mainMenu(allTrips, allStations);
284 }
285
286 /* statsMenu: placeholder for the stats section
287  */
288 void statsMenu() {
289     printf("Sorry, the statistics are not yet implemented\n");
290 }
291
292 /* clearInputBuffer: handles input of multiple characters
293  * resets the input buffer to avoid double-commands
294  * Source:
295  *
stackoverflow.com/questions/3969871/using-getchar-on-c-gets-the-enter-after-input
296  */
297 void clearInputBuffer() // works only if the input buffer is not empty
298 {
299     char c;
300     do {
301         c = getchar();
302     } while (c != '\n' && c != EOF);
303     return;
304 }
```