```c
/**
 * fifteen.c
 *
 * CS50 AP
 * Name: Gonzalo de la Torre Amaya
 *
 * Implements Game of Fifteen (generalized to d x d).
 *
 * Usage: fifteen d
 *
 * whereby the board's dimensions are to be d x d,
 * where d must be in [DIM_MIN,DIM_MAX]
 *
 * Note that usleep is obsolete, but it offers more granularity than
 * sleep and is simpler to use than nanosleep; `man usleep` for more.
 */

// necessary for usleep
#define _XOPEN_SOURCE 500

// libraries to include
#include <cs50.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

// constants
#define DIM_MIN 3
#define DIM_MAX 9
#define BLANK 2424

// globally declared board
int board[DIM_MAX][DIM_MAX];

// globally declared board dimension
int d;

// prototypes
void clear(void);
void greet(void);
void init(void);
void draw(void);
bool move(int tile);
bool won(void);

int main(int argc, string argv[])
{
    // TODO 00: Incorrect usage
    if (argc != 2)
    {
        printf("Usage: fifteen d\n");
        return 1;
    }

    // TODO 01: Be sure that the user puts a dimension 3x3 through 9x9
    d = atoi(argv[1]);
    if (d < DIM_MIN || d > DIM_MAX)
    {
        printf("Board must be between %i x %i and %i x %i, inclusive.\n",
            DIM_MIN, DIM_MIN, DIM_MAX, DIM_MAX);
        return 2;
    }

    // open log file to record moves
    FILE* file = fopen("log.txt", "w");
    if (file == NULL)
    {
        return 3;
    }

    // TODO 02: For the function "greet"
    greet();

    // TODO 03: For the function "init"
    init();

    // accept moves until game is won
    while (true)
    {
        // TODO 04: For the function "clear"
        clear();

        // TODO 05: For the function "draw"
        draw();

        // log the current state of the board (for testing)
        for (int i = 0; i < d; i++)
        {
            for (int j = 0; j < d; j++)
            {
                fprintf(file, "%i", board[i][j]);
                if (j < d - 1)
                {
                    fprintf(file, "|");
                }
            }
            fprintf(file, "\n");
        }
        fflush(file);

        // TODO 06: Congratulate if the user win !
        if (won())
        {
            printf("ftw!\n");
            break;
        }

        // TODO 07: Ask the user for the tile with a GetInt
        printf("Tile to move: ");
        int tile = GetInt();

        // quit if user inputs 0 (for testing)
        if (tile == 0)
        {
            break;
        }

        // log move (for testing)
        fprintf(file, "%i\n", tile);
        fflush(file);

        // TODO 08: If the user does a "illegal move": scold !
        if (!move(tile))
        {
            printf("\nIllegal move.\n");
            usleep(500000);
        }

        // TODO 09: Delay the program 500000 microseconds
        usleep(500000);
    }

    // close log
    fclose(file);

    // TODO 10: To end the game we need to return 0;
    return 0;
}

/**
 * Clears screen using ANSI escape sequences.
 */
void clear(void)
{
```

```c
    printf("\033[2J");
    printf("\033[%d;%dH", 0, 0);
}

/**
 * Greets player.
 */
void greet(void)
{
    clear();
    printf("WELCOME TO GAME OF FIFTEEN\n");
    usleep(2000000);
}

/**
 * Initializes the game's board with tiles numbered 1 through d*d - 1
 * (i.e., fills 2D array with values but does not actually print them).
 */
void init(void)
{
    // Declare the Last number
    int LastNum = d * d - 1;

    for (int i = 0; i < d; i++)
    {

        for(int j = 0; j < d; j++)
        {
            board[i][j] = LastNum;
            LastNum--;
         }

   }

   // Check if the nummber is odd, it must be changed the position of 1 and 2
    if ((d * d - 1) % 2 != 0)
    {
        board[d - 1][d - 2] = 2;
        board[d - 1][d - 3] = 1;
    }
}

/**
 * Prints the board in its current state.
 */
void draw(void)
{
    // TODO
```

```c
     // create board with dimensions given by user

    // For the rows
    for (int i = 0; i < d; i++)
    {
        // To order the numbers in the rows correctly
        for (int j = 0; j < d; j++)
        {
            if (board[i][j] == 0)
            {
                printf("_  ");
            }
            else if (board[i][j] < 10)
            {
                printf("%d  ", board[i][j]);
            }
            else
            {
                printf("%d ", board[i][j]);
            }
        }
        printf("\n");
    }
}

/**
 * If tile borders empty space, moves tile and returns true, else
 * returns false.
 */
bool move(int tile)
{
    // TODO
    for (int i = 0; i < d; i++)
    {
        for (int j = 0; j < d; j++)
        {
            if (board[i][j] == tile)
            {
                if ((i + 1) < d && board[i + 1][j] == 0)
                {
                    board[i + 1][j] = tile;
                    board[i][j] = 0;
                    return true;
                }
                else if ((i - 1) >= 0 && board[i - 1][j] == 0)
                {
                    board[i - 1][j] = tile;
                    board[i][j] = 0;
                    return true;
                }
                else if ((j + 1) < d && board[i][j + 1] == 0)
                {
                    board[i][j + 1] = tile;
                    board[i][j] = 0;
                    return true;
                }
                else if ((j - 1) >= 0 && board[i][j - 1] == 0)
                {
                    board[i][j - 1] = tile;
                    board[i][j] = 0;
                    return true;
                }
            }
        }
    }
    return false;
}

/**
 * Returns true if game is won (i.e., board is in winning configuration),
 * else false.
 */
bool won(void)
{
    int n = 1;
    for (int i = 0; i < d; i++)
    {
        for (int j = 0; j < d; j++)
        {
            if (board[i][j] == n)
            {
                n++;
                if (n == d * d && board[d - 1][d - 1] == 0)
                {
                    return true;
                }
            }
        }
    }
    return false;

}
```