

```
1.  /**
2.   * fifteen.c
3.   *
4.   * CS50 AP
5.   * Name: Andres Joaquin Bolanos Chang
6.   *
7.   * Implements Game of Fifteen (generalized to d x d).
8.   *
9.   * Usage: fifteen d
10.  *
11.  * whereby the board's dimensions are to be d x d,
12.  * where d must be in [DIM_MIN,DIM_MAX]
13.  *
14.  * Note that usleep is obsolete, but it offers more granularity than
15.  * sleep and is simpler to use than nanosleep; `man usleep` for more.
16.  */
17.
18. // necessary for usleep
19. #define _XOPEN_SOURCE 500
20.
21. // libraries to include
22. #include <cs50.h>
23. #include <stdio.h>
24. #include <stdlib.h>
25. #include <unistd.h>
26. #include <string.h>
27. #include <ctype.h>
28.
29. // constants
30. #define DIM_MIN 3
31. #define DIM_MAX 9
32.
33. // globally declared board
34. int board[DIM_MAX][DIM_MAX];
35.
36. // globally declared board dimension
37. int d;
38. int blankspace;
39. int x;
40. int y;
41.
42. // prototypes
43. void clear(void);
44. void greet(void);
45. void init(void);
46. void draw(void);
47. bool move(int tile);
48. bool won(void);
```

```
49.
50. int main(int argc, string argv[])
51. {
52.     // Todo 00: Si hay un numero de comentarios diferente a 2, salirse
53.     if (argc != 2)
54.     {
55.         printf("Usage: fifteen d\n");
56.         return 1;
57.     }
58.
59.     // TODO 01: la dimension en el arg1 deben estar entre MIN y MAX
60.     d = atoi(argv[1]);
61.     if (d < DIM_MIN || d > DIM_MAX)
62.     {
63.         printf("Board must be between %i x %i and %i x %i, inclusive.\n",
64.             DIM_MIN, DIM_MIN, DIM_MAX, DIM_MAX);
65.         return 2;
66.     }
67.
68.     x = (d - 1);
69.     y = (d - 1);
70.
71.     // open log file to record moves
72.     FILE* file = fopen("log.txt", "w");
73.     if (file == NULL)
74.     {
75.         return 3;
76.     }
77.
78.     // TODO 02: El juego te saluda
79.     greet();
80.
81.     // TODO 03: inicializar el "board"
82.     init();
83.
84.     // accept moves until game is won
85.     while (true)
86.     {
87.         // TODO 04: Borra/limpia el terminal
88.         clear();
89.
90.         // TODO 05: Dibuja el "board"
91.         draw();
92.
93.         for (int i = 0; i < d; i++)
94.         {
95.             for (int j = 0; j < d; j++)
96.             {
```

```
97.         fprintf(file, "%i", board[i][j]);
98.         if (j < d - 1)
99.         {
100.             fprintf(file, "|");
101.         }
102.     }
103.     fprintf(file, "\n");
104. }
105. fflush(file);
106.
107. // TODO 06: Checa si ya ganaste
108. if (won())
109. {
110.     printf("ftw!\n");
111.     break;
112. }
113.
114. // TODO 07: Indicar cual "tile" mover
115. printf("Tile to move: ");
116. int tile = GetInt();
117.
118. if (tile == 0)
119. {
120.     break;
121. }
122.
123. fprintf(file, "%i\n", tile);
124. fflush(file);
125.
126. // TODO 08: Si no se puede mover, te dice que es ilegal
127. if (!move(tile))
128. {
129.     printf("\nIllegal move.\n");
130.     usleep(500000);
131. }
132.
133. // TODO 09: Hace que se "duerma" (suspenda) microsegundos el programa
134. usleep(500000);
135. }
136.
137. // close log
138. fclose(file);
139.
140. // TODO 10: Termina el programa
141. return 0;
142. }
143.
144. /**
```

```
145.  * Clears screen using ANSI escape sequences.
146.  */
147.  void clear(void)
148.  {
149.      printf("\033[2J");
150.      printf("\033[%d;%dH", 0, 0);
151.  }
152.
153.  /**
154.   * Greet player.
155.   */
156.  void greet(void)
157.  {
158.      clear();
159.      printf("WELCOME TO GAME OF FIFTEEN\n");
160.      usleep(2000000);
161.  }
162.
163.  /**
164.   * Initializes the game's board with tiles numbered 1 through d*d - 1
165.   * (i.e., fills 2D array with values but does not actually print them).
166.   */
167.  void init(void)
168.  {
169.      int spacenum = (d * d) - 1;
170.      for (int i = 0; i < d; i++)
171.      {
172.          for (int j = 0; j < d; j++)
173.          {
174.              board[i][j] = spacenum;
175.              spacenum = spacenum - 1;
176.          }
177.      }
178.      if ((d % 2) == 0)
179.      {
180.          int temp = board[d - 1][d - 2];
181.          board[d - 1][d - 2] = board[d - 1][d - 3];
182.          board[d - 1][d - 3] = temp;
183.      }
184.  }
185.
186.  /**
187.   * Prints the board in its current state.
188.   */
189.  void draw(void)
190.  {
191.      for (int i = 0; i < d; i++)
192.      {
```

```
193.         for (int j = 0; j < d; j++)
194.         {
195.             if (board[i][j] < 10)
196.             {
197.                 if (board[i][j] == 0)
198.                 {
199.                     printf(" _ ");
200.                 }
201.                 else
202.                 {
203.                     printf(" %d ", board[i][j]);
204.                 }
205.             }
206.             else
207.             {
208.                 printf("%d ", board[i][j]);
209.             }
210.         }
211.         printf("\n\n");
212.     }
213. }
214.
215. /**
216.  * If tile borders empty space, moves tile and returns true, else
217.  * returns false.
218.  */
219. bool move(int tile)
220. {
221.     for (int i = 0; i < d; i++)
222.     {
223.         for (int j = 0; j < d; j++)
224.         {
225.             if (tile == board[i][j])
226.             {
227.                 int blankspace = 0;
228.
229.                 if (((x == (i - 1)) && (j == y)) || ((i == x) && (y == (j + 1))
230.                 ) || ((i == x) && (y == (j - 1))) || ((x == (i + 1)) && (j == y)
231.                 ))
232.                 {
233.                     board[x][y] = tile;
234.                     board[i][j] = blankspace;
235.                     x = i;
236.                     y = j;
237.                     printf("\n");
238.                     return true;
239.                 }
240.             }
```

```
241.     }
242. }
243. return false;
244. }
245.
246. /**
247.  * Returns true if game is won (i.e., board is in winning configuration),
248.  * else false.
249.  */
250. bool won(void)
251. {
252.     int n = 1;
253.     // iterates through board
254.     for (int i = 0; i < d; i++)
255.     {
256.         for (int j = 0; j < d; j++)
257.         {
258.             // if any tile != counter, counting from 0, return false
259.             if (board[i][j] == n)
260.             {
261.                 n++;
262.                 if (n == (d * d) && board[d - 1][d - 1] == 0)
263.                 {
264.                     return true;
265.                 }
266.             }
267.         }
268.     }
269.     return false;
270. }
```