```
#
# Makefile
#
# Computer Science 50
# Problem Set 3
#

all: find generate

find: find.c helpers.c helpers.h
    gcc -ggdb -std=c99 -Wall -Werror -Wformat=0 -o find find.c helpers.c -lcs50 -lm

generate: generate.c
    gcc -ggdb -std=c99 -Wall -Werror -Wformat=0 -o generate generate.c

clean:
    rm -f *.o a.out core find generate

```

```c
/**************************************************************************
 * find.c
 *
 * Computer Science 50
 * Problem Set 3
 *
 * Prompts user for as many as HAY_MAX values until EOF is reached,
 * then proceeds to search that "haystack" of values for given needle.
 *
 * Usage: find needle
 *
 * where needle is the value to find in a haystack of values
 **************************************************************************/

#include <cs50.h>
#include <stdio.h>
#include <stdlib.h>

#include "helpers.h"


// maximum amount of hay
const int HAY_MAX = 65536;


int
main(int argc, char *argv[])
{
    // ensure proper usage
    if (argc != 2)
    {
        printf("Usage: %s needle\n", argv[0]);
        return 1;
    }

    // remember needle
    int needle = atoi(argv[1]);

    // fill haystack
    int size;
    int haystack[HAY_MAX];
    for (size = 0; size < HAY_MAX; size++)
    {
        // wait for hay until EOF
        printf("\nhaystack[%d] = ", size);
        int straw = GetInt();
        if (straw == INT_MAX)
            break;
```

```c

        // add hay to stack
        haystack[size] = straw;
    }
    printf("\n");

    // sort the haystack
    sort(haystack, size);

    // try to find needle in haystack
    if (search(needle, haystack, size))
        printf("\nFound needle in haystack!\n\n");
    else
        printf("\nDidn't find needle in haystack.\n\n");

    // that's all folks
    return 0;
}

```

```c
/**************************************************************************
 * generate.c
 *
 * Computer Science 50
 * Problem Set 3
 *
 * Generates pseudorandom numbers in [0,LIMIT), one per line.
 *
 * Usage: generate n [s]
 *
 * where n is number of pseudorandom numbers to print
 * and s is an optional seed
 **************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define LIMIT 65536

int
main(int argc, char *argv[])
{
    // TODO: comment me
    if (argc != 2 && argc != 3)
    {
        printf("Usage: %s n [s]\n", argv[0]);
        return 1;
    }

    // TODO: comment me
    int n = atoi(argv[1]);

    // TODO: comment me
    if (argc == 3)
        srand((unsigned int) atoi(argv[2]));
    else
        srand((unsigned int) time(NULL));

    // TODO: comment me
    for (int i = 0; i < n; i++)
        printf("%d\n", rand() % LIMIT);

    // that's all folks
    return 0;
}

```

```c
/**************************************************************************
 * helpers.c
 *
 * Computer Science 50
 * Problem Set 3
 *
 * Helper functions for Problem Set 3.
 **************************************************************************/

#include <cs50.h>

#include "helpers.h"


/*
 * Returns true if value is in array of n values, else false.
 */

bool
search(int value, int array[], int n)
{
    // TODO: re-implement as binary search
    for (int i = 0; i < n; i++)
        if (array[i] == value)
            return true;
    return false;
}


/*
 * Sorts array of n values.
 */

void
sort(int values[], int n)
{
    // TODO: implement an O(n^2) sort
    return;
}

```

```c
/**************************************************************************
 * helpers.h
 *
 * Computer Science 50
 * Problem Set 3
 *
 * Helper functions for Problem Set 3.
 **************************************************************************/

#include <cs50.h>


/*
 * Returns true if value is in array of n values, else false.
 */

bool
search(int value, int values[], int n);


/*
 * Sorts array of n values.
 */

void
sort(int values[], int n);

```

```
#
# Makefile
#
# Computer Science 50
# Problem Set 3
#

fifteen: fifteen.c
    gcc -ggdb -std=c99 -Wall -Werror -Wformat=0 -o fifteen fifteen.c -lcs50 -lm

clean:
    rm -f *.o a.out core fifteen

```

```c
/***************************************************************************
 * fifteen.c
 *
 * Computer Science 50
 * Problem Set 3
 *
 * Implements The 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.
 ***************************************************************************/

#define _XOPEN_SOURCE 500

#include <cs50.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


// constants
#define DIM_MIN 3
#define DIM_MAX 9


// board
int board[DIM_MAX][DIM_MAX];

// dimensions
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, char *argv[])
```

```
49. {
50.     // greet user with instructions
51.     greet();
52. 
53.     // ensure proper usage
54.     if (argc != 2)
55.     {
56.         printf("Usage: %s d\n", argv[0]);
57.         return 1;
58.     }
59. 
60.     // ensure valid dimensions
61.     d = atoi(argv[1]);
62.     if (d < DIM_MIN || d > DIM_MAX)
63.     {
64.         printf("Board must be between %d x %d and %d x %d, inclusive.\n",
65.          DIM_MIN, DIM_MIN, DIM_MAX, DIM_MAX);
66.         return 2;
67.     }
68. 
69.     // initialize the board
70.     init();
71. 
72.     // accept moves until game is won
73.     while (true)
74.     {
75.         // clear the screen
76.         clear();
77. 
78.         // draw the current state of the board
79.         draw();
80. 
81.         // check for win
82.         if (won())
83.         {
84.             printf("ftw!\n");
85.             break;
86.         }
87. 
88.         // prompt for move
89.         printf("Tile to move: ");
90.         int tile = GetInt();
91. 
92.         // move if possible, else report illegality
93.         if (!move(tile))
94.         {
95.             printf("\nIllegal move.\n");
96.             usleep(500000);
```

```
 97.         }
 98.
 99.         // sleep thread for animation's sake
100.         usleep(500000);
101.     }
102.
103.     // that's all folks
104.     return 0;
105. }
106.
107.
108. /*
109.  * Clears screen using ANSI escape sequences.
110.  */
111.
112. void
113. clear(void)
114. {
115.     printf("\033[2J");
116.     printf("\033[%d;%dH", 0, 0);
117. }
118.
119.
120. /*
121.  * Greets player.
122.  */
123.
124. void
125. greet(void)
126. {
127.     clear();
128.     printf("WELCOME TO THE GAME OF FIFTEEN\n");
129.     usleep(2000000);
130. }
131.
132.
133. /*
134.  * Initializes the game's board with tiles numbered 1 through d*d - 1
135.  * (i.e., fills 2D array with values but does not actually print them).
136.  */
137.
138. void
139. init(void)
140. {
141.     // TODO
142. }
143.
144.
```

```c
/*
 * Prints the board in its current state.
 */

void
draw(void)
{
    // TODO
}


/*
 * If tile borders empty space, moves tile and returns true, else
 * returns false.
 */

bool
move(int tile)
{
    // TODO
    return false;
}


/*
 * Returns true if game is won (i.e., board is in winning configuration),
 * else false.
 */

bool
won(void)
{
    // TODO
    return false;
}

```