```c
/**************************************************************************
 * argv1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Prints command-line arguments, one per line.
 *
 * Demonstrates use of argv.
 **************************************************************************/

#include <stdio.h>


int
main(int argc, char *argv[])
{
    // print arguments
    printf("\n");
    for (int i = 0; i < argc; i++)
        printf("%s\n", argv[i]);
    printf("\n");
}
```

```c
/**************************************************************************
 * argv2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Prints command-line arguments, one character per line.
 *
 * Demonstrates argv as a two-dimensional array.
 **************************************************************************/

#include <stdio.h>
#include <string.h>


int
main(int argc, char *argv[])
{
    // print arguments
    printf("\n");
    for (int i = 0; i < argc; i++)
    {
        for (int j = 0, n = strlen(argv[i]); j < n; j++)
            printf("%c\n", argv[i][j]);
        printf("\n");
    }
}
```

```c
/**************************************************************************
 * array1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Computes a student's average across 2 quizzes.
 *
 * Demonstrates C's math library.
 **************************************************************************/

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


// number of quizzes per term
#define QUIZZES 2


int
main(void)
{
    // ask user for grades
    float grades[QUIZZES];
    printf("\nWhat were your quiz scores?\n\n");
    for (int i = 0; i < QUIZZES; i++)
    {
        printf("Quiz #%d of %d: ", i+1, QUIZZES);
        grades[i] = GetFloat();
    }

    // compute average
    float sum = 0;
    for (int i = 0; i < QUIZZES; i++)
        sum += grades[i];
    int average = (int) round(sum / QUIZZES);

    // report average
    printf("\nYour average is: %d\n\n", average);
}
```

```c
/**************************************************************************
 * array2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Computes a student's average across 2 quizzes.
 *
 * Demonstrates use of an array, a constant, and rounding.
 **************************************************************************/

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


// number of quizzes per term
#define QUIZZES 2


int
main(void)
{
    // ask user for grades
    float grades[QUIZZES];
    printf("\nWhat were your quiz scores?\n\n");
    for (int i = 0; i < QUIZZES; i++)
    {
        printf("Quiz #%d of %d: ", i+1, QUIZZES);
        grades[i] = GetFloat();
    }

    // compute average
    float sum = 0;
    for (int i = 0; i < QUIZZES; i++)
        sum += grades[i];
    int average = (int) (sum / QUIZZES + 0.5);

    // report average
    printf("\nYour average is: %d\n\n", average);
}
```

```c
/**************************************************************************
 * beer1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Sings "99 Bottles of Beer on the Wall."
 *
 * Demonstrates a for loop (and an opportunity for hierarchical
 * decomposition).
 **************************************************************************/

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


int
main(void)
{
    // ask user for number
    printf("How many bottles will there be? ");
    int n = GetInt();

    // exit upon invalid input
    if (n < 1)
    {
        printf("Sorry, that makes no sense.\n");
        return 1;
    }

    // sing the annoying song
    printf("\n");
    for (int i = n; i > 0; i--)
    {
        printf("%d bottle(s) of beer on the wall,\n", i);
        printf("%d bottle(s) of beer,\n", i);
        printf("Take one down, pass it around,\n");
        printf("%d bottle(s) of beer on the wall.\n\n", i - 1);
    }

    // exit when song is over
    printf("Wow, that's annoying.\n");
    return 0;
}
```

```c
/**************************************************************************
 * beer2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Sings "99 Bottles of Beer on the Wall."
 *
 * Demonstrates a while loop (and an opportunity for hierarchical
 * decomposition).
 **************************************************************************/

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


int
main(void)
{
    // ask user for number
    printf("How many bottles will there be? ");
    int n = GetInt();

    // exit upon invalid input
    if (n < 1)
    {
        printf("Sorry, that makes no sense.\n");
        return 1;
    }

    // sing the annoying song
    printf("\n");
    while (n > 0)
    {
        printf("%d bottle(s) of beer on the wall,\n", n);
        printf("%d bottle(s) of beer,\n", n);
        printf("Take one down, pass it around,\n");
        printf("%d bottle(s) of beer on the wall.\n\n", n - 1);
        n--;
    }

    // exit when song is over
    printf("Wow, that's annoying.\n");
    return 0;
}
```

```c
/**************************************************************************
 * beer3.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Sings "99 Bottles of Beer on the Wall."
 *
 * Demonstrates a condition within a for loop.
 **************************************************************************/

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


int
main(void)
{
    // ask user for number
    printf("How many bottles will there be? ");
    int n = GetInt();

    // exit upon invalid input
    if (n < 1)
    {
        printf("Sorry, that makes no sense.\n");
        return 1;
    }

    // sing the annoying song
    printf("\n");
    for (int i = n; i > 0; i--)
    {
        // use proper grammar
        string s1 = (i == 1) ? "bottle" : "bottles";
        string s2 = (i == 2) ? "bottle" : "bottles";

        // sing verses
        printf("%d %s of beer on the wall,\n", i, s1);
        printf("%d %s of beer,\n", i, s1);
        printf("Take one down, pass it around,\n");
        printf("%d %s of beer on the wall.\n\n", i - 1, s2);
    }

    // exit when song is over
    printf("Wow, that's annoying.\n");
    return 0;
}
```

```c
/**************************************************************************
 * beer4.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Sings "99 Bottles of Beer on the Wall."
 *
 * Demonstrates hierarchical decomposition and parameter passing.
 **************************************************************************/

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


// function prototype
void chorus(int b);


int
main(void)
{
    // ask user for number
    printf("How many bottles will there be? ");
    int n = GetInt();

    // exit upon invalid input
    if (n < 1)
    {
        printf("Sorry, that makes no sense.\n");
        return 1;
    }

    // sing the annoying song
    printf("\n");
    while (n)
        chorus(n--);

    // exit when song is over
    printf("Wow, that's annoying.\n");
    return 0;
}


/*
 * Sings about specified number of bottles.
 */

```

```c
void
chorus(int b)
{
    // use proper grammar
    string s1 = (b == 1) ? "bottle" : "bottles";
    string s2 = (b == 2) ? "bottle" : "bottles";

    // sing verses
    printf("%d %s of beer on the wall,\n", b, s1);
    printf("%d %s of beer,\n", b, s1);
    printf("Take one down, pass it around,\n");
    printf("%d %s of beer on the wall.\n\n", b - 1, s2);
}
```

```c
/**************************************************************************
 * buggy3.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Should swap two variables' values, but doesn't!
 * Can you find the bug?
 **************************************************************************/

#include <stdio.h>


// function prototype
void swap(int a, int b);


int
main(void)
{
    int x = 1;
    int y = 2;

    printf("x is %d\n", x);
    printf("y is %d\n", y);
    printf("Swapping...\n");
    swap(x, y);
    printf("Swapped!\n");
    printf("x is %d\n", x);
    printf("y is %d\n", y);
}


/*
 * Swap arguments' values.
 */

void
swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

```c
/**************************************************************************
 * buggy4.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Should increment a variable, but doesn't!
 * Can you find the bug?
 **************************************************************************/

#include <stdio.h>


// function prototype
void increment(void);


int
main(void)
{
    int x = 1;
    printf("x is now %d\n", x);
    printf("Incrementing...\n");
    increment();
    printf("Incremented!\n");
    printf("x is now %d\n", x);
}


/*
 * Tries to increment x.
 */

void
increment(void)
{
    x++;
}
```

```c
/**************************************************************************
 * buggy5.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Should increment a variable, but doesn't!
 * Can you find the bug?
 **************************************************************************/

#include <stdio.h>


// global variable
int x;

// function prototype
void increment(void);


int
main(void)
{
    printf("x is now %d\n", x);
    printf("Initializing...\n");
    x = 1;
    printf("Initialized!\n");
    printf("x is now %d\n", x);
    printf("Incrementing...\n");
    increment();
    printf("Incremented!\n");
    printf("x is now %d\n", x);
}


/*
 * Increments x.
 */

void
increment(void)
{
    int x = 10;
    x++;
}
```

```c
/**************************************************************************
 * buggy6.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Asks student for their grades but prints too many!
 * Can you find the bug?
 *
 * Demonstrates accidental use of a "magic number."
 **************************************************************************/

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


// number of quizzes per term
#define QUIZZES 2


int
main(void)
{
    float grades[QUIZZES];

    // ask user for scores
    printf("\nWhat were your quiz scores?\n\n");
    for (int i = 0; i < QUIZZES; i++)
    {
        printf("Quiz #%d of %d: ", i+1, QUIZZES);
        grades[i] = GetFloat();
    }

    // print scores
    for (int i = 0; i < 3; i++)
        printf("%.2f\n", grades[i]);
}
```

```
/**************************************************************************
 * CS50 Library 3.0
 *
 * https://manual.cs50.net/Library
 *
 * Glenn Holloway <holloway@eecs.harvard.edu>
 * David J. Malan <malan@harvard.edu>
 *
 * Based on Eric Roberts' genlib.c and simpio.c.
 *
 * Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
 * http://creativecommons.org/licenses/by-nc-sa/3.0/
 **************************************************************************/

#ifndef _CS50_H
#define _CS50_H

#include <float.h>
#include <limits.h>
#include <stdbool.h>
#include <stdlib.h>


/*
 * Our own data type for string variables.
 */

typedef char *string;


/*
 * Reads a line of text from standard input and returns the equivalent
 * char; if text does not represent a char, user is prompted to retry.
 * Leading and trailing whitespace is ignored.  If line can't be read,
 * returns CHAR_MAX.
 */

char
GetChar(void);


/*
 * Reads a line of text from standard input and returns the equivalent
 * double as precisely as possible; if text does not represent a
 * double, user is prompted to retry.  Leading and trailing whitespace
 * is ignored.  For simplicity, overflow and underflow are not detected.
 * If line can't be read, returns DBL_MAX.
 */
```

```
49.
50. double
51. GetDouble(void);
52.
53.
54. /*
55.  * Reads a line of text from standard input and returns the equivalent
56.  * float as precisely as possible; if text does not represent a float,
57.  * user is prompted to retry.  Leading and trailing whitespace is ignored.
58.  * For simplicity, overflow and underflow are not detected.  If line can't
59.  * be read, returns FLT_MAX.
60.  */
61.
62. float
63. GetFloat(void);
64.
65.
66. /*
67.  * Reads a line of text from standard input and returns it as an
68.  * int in the range of [-2^31 + 1, 2^31 - 2], if possible; if text
69.  * does not represent such an int, user is prompted to retry.  Leading
70.  * and trailing whitespace is ignored.  For simplicity, overflow is not
71.  * detected.  If line can't be read, returns INT_MAX.
72.  */
73.
74. int
75. GetInt(void);
76.
77.
78. /*
79.  * Reads a line of text from standard input and returns an equivalent
80.  * long long in the range [-2^63 + 1, 2^63 - 2], if possible; if text
81.  * does not represent such a long long, user is prompted to retry.
82.  * Leading and trailing whitespace is ignored.  For simplicity, overflow
83.  * is not detected.  If line can't be read, returns LLONG_MAX.
84.  */
85.
86. long long
87. GetLongLong(void);
88.
89.
90. /*
91.  * Reads a line of text from standard input and returns it as a
92.  * string (char *), sans trailing newline character.  (Ergo, if
93.  * user inputs only "\n", returns "" not NULL.)  Returns NULL
94.  * upon error or no input whatsoever (i.e., just EOF).  Leading
95.  * and trailing whitespace is not ignored.  Stores string on heap
96.  * (via malloc); memory must be freed by caller to avoid leak.
```

```
 */

string GetString(void);


#endif
```

```c
/**********************************************************************
 * sigma1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Adds the numbers 1 through n.
 *
 * Demonstrates iteration.
 **********************************************************************/

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


// prototype
int sigma(int);


int
main(void)
{
    // ask user for a positive int
    int n;
    do
    {
        printf("Positive integer please: ");
        n = GetInt();
    }
    while (n < 1);

    // compute sum of 1 through n
    int answer = sigma(n);

    // report answer
    printf("%d\n", answer);
}


/*
 * Returns sum of 1 through m; returns 0 if m is not positive.
 */

int
sigma(int m)
{
    // avoid risk of infinite loop
    if (m < 1)
```

```c
        return 0;

    // return sum of 1 through m
    int sum = 0;
    for (int i = 1; i <= m; i++)
        sum += i;
    return sum;
}
```

```c
/**********************************************************************
 * sigma2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Adds the numbers 1 through n.
 *
 * Demonstrates recursion.
 **********************************************************************/

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


// prototype
int sigma(int);


int
main(void)
{
    // ask user for a positive int
    int n;
    do
    {
        printf("Positive integer please: ");
        n = GetInt();
    }
    while (n < 1);

    // compute sum of 1 through n
    int answer = sigma(n);

    // report answer
    printf("%d\n", answer);
}


/*
 * Returns sum of 1 through m; returns 0 if m is not positive.
 */

int
sigma(int m)
{
    // base case
    if (m <= 0)
```

```
49.         return 0;
50.
51.     // recursive case
52.     else
53.         return (m + sigma(m-1));
54. }
```

```c
/****************************************************************************
 * string1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Prints a given string one character per line.
 *
 * Demonstrates strings as arrays of chars and use of strlen.
 ****************************************************************************/

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


int
main(void)
{
    // get line of text
    string s = GetString();

    // print string, one character per line
    if (s != NULL)
    {
        for (int i = 0; i < strlen(s); i++)
        {
            char c = s[i];
            printf("%c\n", c);
        }
    }
}
```

```c
/**************************************************************************
 * string2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Prints a given string one character per line.
 *
 * Demonstrates strings as arrays of chars with slight optimization.
 **************************************************************************/

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


int
main(void)
{
    // get line of text
    string s = GetString();

    // print string, one character per line
    if (s != NULL)
    {
        for (int i = 0, n = strlen(s); i < n; i++)
        {
            printf("%c\n", s[i]);
        }
    }
}
```