

```
1.  /*****
2.   * argv1.c
3.   *
4.   * Computer Science 50
5.   * David J. Malan
6.   *
7.   * Prints command-line arguments, one per line.
8.   *
9.   * Demonstrates use of argv.
10.  *****/
11.
12. #include <stdio.h>
13.
14.
15. int
16. main(int argc, char *argv[])
17. {
18.     // print arguments
19.     printf("\n");
20.     for (int i = 0; i < argc; i++)
21.         printf("%s\n", argv[i]);
22.     printf("\n");
23. }
```

```
1. /*****
2.  * argv2.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Prints command-line arguments, one character per line.
8.  *
9.  * Demonstrates argv as a two-dimensional array.
10. *****/
11.
12. #include <stdio.h>
13. #include <string.h>
14.
15.
16. int
17. main(int argc, char *argv[])
18. {
19.     // print arguments
20.     printf("\n");
21.     for (int i = 0; i < argc; i++)
22.     {
23.         for (int j = 0, n = strlen(argv[i]); j < n; j++)
24.             printf("%c\n", argv[i][j]);
25.         printf("\n");
26.     }
27. }
```

```
1.  /*****
2.  * array1.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Computes a student's average across 2 quizzes.
8.  *
9.  * Demonstrates C's math library.
10. *****/
11.
12. #include <cs50.h>
13. #include <math.h>
14. #include <stdio.h>
15.
16.
17. // number of quizzes per term
18. #define QUIZZES 2
19.
20.
21. int
22. main(void)
23. {
24.     // ask user for grades
25.     float grades[QUIZZES];
26.     printf("\nWhat were your quiz scores?\n\n");
27.     for (int i = 0; i < QUIZZES; i++)
28.     {
29.         printf("Quiz #%d of %d: ", i+1, QUIZZES);
30.         grades[i] = GetFloat();
31.     }
32.
33.     // compute average
34.     float sum = 0;
35.     for (int i = 0; i < QUIZZES; i++)
36.         sum += grades[i];
37.     int average = (int) round(sum / QUIZZES);
38.
39.     // report average
40.     printf("\nYour average is: %d\n\n", average);
41. }
```

```
1.  /*****
2.  * array2.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Computes a student's average across 2 quizzes.
8.  *
9.  * Demonstrates use of an array, a constant, and rounding.
10. *****/
11.
12. #include <cs50.h>
13. #include <stdio.h>
14.
15.
16. // number of quizzes per term
17. #define QUIZZES 2
18.
19.
20. int
21. main(void)
22. {
23.     // ask user for grades
24.     float grades[QUIZZES];
25.     printf("\nWhat were your quiz scores?\n\n");
26.     for (int i = 0; i < QUIZZES; i++)
27.     {
28.         printf("Quiz #%d of %d: ", i+1, QUIZZES);
29.         grades[i] = GetFloat();
30.     }
31.
32.     // compute average
33.     float sum = 0;
34.     for (int i = 0; i < QUIZZES; i++)
35.         sum += grades[i];
36.     int average = (int) (sum / QUIZZES + 0.5);
37.
38.     // report average
39.     printf("\nYour average is: %d\n\n", average);
40. }
```

```
1. /*****
2.  * beer1.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Sings "99 Bottles of Beer on the Wall."
8.  *
9.  * Demonstrates a for loop (and an opportunity for hierarchical
10. * decomposition).
11. *****/
12.
13. #include <cs50.h>
14. #include <stdio.h>
15.
16.
17. int
18. main(void)
19. {
20.     // ask user for number
21.     printf("How many bottles will there be? ");
22.     int n = GetInt();
23.
24.     // exit upon invalid input
25.     if (n < 1)
26.     {
27.         printf("Sorry, that makes no sense.\n");
28.         return 1;
29.     }
30.
31.     // sing the annoying song
32.     printf("\n");
33.     for (int i = n; i > 0; i--)
34.     {
35.         printf("%d bottle(s) of beer on the wall,\n", i);
36.         printf("%d bottle(s) of beer,\n", i);
37.         printf("Take one down, pass it around,\n");
38.         printf("%d bottle(s) of beer on the wall.\n\n", i - 1);
39.     }
40.
41.     // exit when song is over
42.     printf("Wow, that's annoying.\n");
43.     return 0;
44. }
```

```
1. /*****
2.  * beer2.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Sings "99 Bottles of Beer on the Wall."
8.  *
9.  * Demonstrates a while loop (and an opportunity for hierarchical
10. * decomposition).
11. *****/
12.
13. #include <cs50.h>
14. #include <stdio.h>
15.
16.
17. int
18. main(void)
19. {
20.     // ask user for number
21.     printf("How many bottles will there be? ");
22.     int n = GetInt();
23.
24.     // exit upon invalid input
25.     if (n < 1)
26.     {
27.         printf("Sorry, that makes no sense.\n");
28.         return 1;
29.     }
30.
31.     // sing the annoying song
32.     printf("\n");
33.     while (n > 0)
34.     {
35.         printf("%d bottle(s) of beer on the wall,\n", n);
36.         printf("%d bottle(s) of beer,\n", n);
37.         printf("Take one down, pass it around,\n");
38.         printf("%d bottle(s) of beer on the wall.\n\n", n - 1);
39.         n--;
40.     }
41.
42.     // exit when song is over
43.     printf("Wow, that's annoying.\n");
44.     return 0;
45. }
```

```
1. /*****
2.  * beer3.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Sings "99 Bottles of Beer on the Wall."
8.  *
9.  * Demonstrates a condition within a for loop.
10. *****/
11.
12. #include <cs50.h>
13. #include <stdio.h>
14.
15.
16. int
17. main(void)
18. {
19.     // ask user for number
20.     printf("How many bottles will there be? ");
21.     int n = GetInt();
22.
23.     // exit upon invalid input
24.     if (n < 1)
25.     {
26.         printf("Sorry, that makes no sense.\n");
27.         return 1;
28.     }
29.
30.     // sing the annoying song
31.     printf("\n");
32.     for (int i = n; i > 0; i--)
33.     {
34.         // use proper grammar
35.         string s1 = (i == 1) ? "bottle" : "bottles";
36.         string s2 = (i == 2) ? "bottle" : "bottles";
37.
38.         // sing verses
39.         printf("%d %s of beer on the wall,\n", i, s1);
40.         printf("%d %s of beer,\n", i, s1);
41.         printf("Take one down, pass it around,\n");
42.         printf("%d %s of beer on the wall.\n\n", i - 1, s2);
43.     }
44.
45.     // exit when song is over
46.     printf("Wow, that's annoying.\n");
47.     return 0;
48. }
```

```
1. /*****
2.  * beer4.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Sings "99 Bottles of Beer on the Wall."
8.  *
9.  * Demonstrates hierarchical decomposition and parameter passing.
10. *****/
11.
12. #include <cs50.h>
13. #include <stdio.h>
14.
15.
16. // function prototype
17. void chorus(int b);
18.
19.
20. int
21. main(void)
22. {
23.     // ask user for number
24.     printf("How many bottles will there be? ");
25.     int n = GetInt();
26.
27.     // exit upon invalid input
28.     if (n < 1)
29.     {
30.         printf("Sorry, that makes no sense.\n");
31.         return 1;
32.     }
33.
34.     // sing the annoying song
35.     printf("\n");
36.     while (n)
37.         chorus(n--);
38.
39.     // exit when song is over
40.     printf("Wow, that's annoying.\n");
41.     return 0;
42. }
43.
44.
45. /*
46.  * Sings about specified number of bottles.
47.  */
48.
```



```
49. void
50. chorus(int b)
51. {
52.     // use proper grammar
53.     string s1 = (b == 1) ? "bottle" : "bottles";
54.     string s2 = (b == 2) ? "bottle" : "bottles";
55.
56.     // sing verses
57.     printf("%d %s of beer on the wall,\n", b, s1);
58.     printf("%d %s of beer,\n", b, s1);
59.     printf("Take one down, pass it around,\n");
60.     printf("%d %s of beer on the wall.\n\n", b - 1, s2);
61. }
```

```
1. /*****
2.  * buggy3.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Should swap two variables' values, but doesn't!
8.  * Can you find the bug?
9.  *****/
10.
11. #include <stdio.h>
12.
13.
14. // function prototype
15. void swap(int a, int b);
16.
17.
18. int
19. main(void)
20. {
21.     int x = 1;
22.     int y = 2;
23.
24.     printf("x is %d\n", x);
25.     printf("y is %d\n", y);
26.     printf("Swapping...\n");
27.     swap(x, y);
28.     printf("Swapped!\n");
29.     printf("x is %d\n", x);
30.     printf("y is %d\n", y);
31. }
32.
33.
34. /*
35.  * Swap arguments' values.
36.  */
37.
38. void
39. swap(int a, int b)
40. {
41.     int tmp = a;
42.     a = b;
43.     b = tmp;
44. }
```

```
1.  /*****
2.   * buggy4.c
3.   *
4.   * Computer Science 50
5.   * David J. Malan
6.   *
7.   * Should increment a variable, but doesn't!
8.   * Can you find the bug?
9.   *****/
10.
11. #include <stdio.h>
12.
13.
14. // function prototype
15. void increment(void);
16.
17.
18. int
19. main(void)
20. {
21.     int x = 1;
22.     printf("x is now %d\n", x);
23.     printf("Incrementing...\n");
24.     increment();
25.     printf("Incremented!\n");
26.     printf("x is now %d\n", x);
27. }
28.
29.
30. /*
31.  * Tries to increment x.
32.  */
33.
34. void
35. increment(void)
36. {
37.     x++;
38. }
```

```
1.  /*****
2.   * buggy5.c
3.   *
4.   * Computer Science 50
5.   * David J. Malan
6.   *
7.   * Should increment a variable, but doesn't!
8.   * Can you find the bug?
9.   *****/
10.
11. #include <stdio.h>
12.
13.
14. // global variable
15. int x;
16.
17. // function prototype
18. void increment(void);
19.
20.
21. int
22. main(void)
23. {
24.     printf("x is now %d\n", x);
25.     printf("Initializing...\n");
26.     x = 1;
27.     printf("Initialized!\n");
28.     printf("x is now %d\n", x);
29.     printf("Incrementing...\n");
30.     increment();
31.     printf("Incremented!\n");
32.     printf("x is now %d\n", x);
33. }
34.
35.
36. /*
37.  * Increments x.
38.  */
39.
40. void
41. increment(void)
42. {
43.     int x = 10;
44.     x++;
45. }
```

```
1. /*****
2.  * buggy6.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Asks student for their grades but prints too many!
8.  * Can you find the bug?
9.  *
10. * Demonstrates accidental use of a "magic number."
11. *****/
12.
13. #include <cs50.h>
14. #include <stdio.h>
15.
16.
17. // number of quizzes per term
18. #define QUIZZES 2
19.
20.
21. int
22. main(void)
23. {
24.     float grades[QUIZZES];
25.
26.     // ask user for scores
27.     printf("\nWhat were your quiz scores?\n\n");
28.     for (int i = 0; i < QUIZZES; i++)
29.     {
30.         printf("Quiz #%d of %d: ", i+1, QUIZZES);
31.         grades[i] = GetFloat();
32.     }
33.
34.     // print scores
35.     for (int i = 0; i < 3; i++)
36.         printf("%.2f\n", grades[i]);
37. }
```

```
1. /*****
2.  * CS50 Library 3.0
3.  *
4.  * https://manual.cs50.net/Library
5.  *
6.  * Glenn Holloway <holloway@eecs.harvard.edu>
7.  * David J. Malan <malan@harvard.edu>
8.  *
9.  * Based on Eric Roberts' genlib.c and simpio.c.
10. *
11. * Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
12. * http://creativecommons.org/licenses/by-nc-sa/3.0/
13. *****/
14.
15. #ifndef _CS50_H
16. #define _CS50_H
17.
18. #include <float.h>
19. #include <limits.h>
20. #include <stdbool.h>
21. #include <stdlib.h>
22.
23.
24. /*
25.  * Our own data type for string variables.
26.  */
27.
28. typedef char *string;
29.
30.
31. /*
32.  * Reads a line of text from standard input and returns the equivalent
33.  * char; if text does not represent a char, user is prompted to retry.
34.  * Leading and trailing whitespace is ignored. If line can't be read,
35.  * returns CHAR_MAX.
36.  */
37.
38. char
39. GetChar(void);
40.
41.
42. /*
43.  * Reads a line of text from standard input and returns the equivalent
44.  * double as precisely as possible; if text does not represent a
45.  * double, user is prompted to retry. Leading and trailing whitespace
46.  * is ignored. For simplicity, overflow and underflow are not detected.
47.  * If line can't be read, returns DBL_MAX.
48.  */
```

```
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.
```

---

```
97.  */
98.
99.  string GetString(void);
100.
101.
102.
103. #endif
```



```
1.  /*****
2.  *  sigma1.c
3.  *
4.  *  Computer Science 50
5.  *  David J. Malan
6.  *
7.  *  Adds the numbers 1 through n.
8.  *
9.  *  Demonstrates iteration.
10. *****/
11.
12. #include <cs50.h>
13. #include <stdio.h>
14.
15.
16. // prototype
17. int sigma(int);
18.
19.
20. int
21. main(void)
22. {
23.     // ask user for a positive int
24.     int n;
25.     do
26.     {
27.         printf("Positive integer please: ");
28.         n = GetInt();
29.     }
30.     while (n < 1);
31.
32.     // compute sum of 1 through n
33.     int answer = sigma(n);
34.
35.     // report answer
36.     printf("%d\n", answer);
37. }
38.
39.
40. /*
41. *  Returns sum of 1 through m; returns 0 if m is not positive.
42. */
43.
44. int
45. sigma(int m)
46. {
47.     // avoid risk of infinite loop
48.     if (m < 1)
```

```
49.     return 0;
50.
51.     // return sum of 1 through m
52.     int sum = 0;
53.     for (int i = 1; i <= m; i++)
54.         sum += i;
55.     return sum;
56. }
```

```
1.  /*****
2.  *  sigma2.c
3.  *
4.  *  Computer Science 50
5.  *  David J. Malan
6.  *
7.  *  Adds the numbers 1 through n.
8.  *
9.  *  Demonstrates recursion.
10. *****/
11.
12. #include <cs50.h>
13. #include <stdio.h>
14.
15.
16. // prototype
17. int sigma(int);
18.
19.
20. int
21. main(void)
22. {
23.     // ask user for a positive int
24.     int n;
25.     do
26.     {
27.         printf("Positive integer please: ");
28.         n = GetInt();
29.     }
30.     while (n < 1);
31.
32.     // compute sum of 1 through n
33.     int answer = sigma(n);
34.
35.     // report answer
36.     printf("%d\n", answer);
37. }
38.
39.
40. /*
41. *  Returns sum of 1 through m; returns 0 if m is not positive.
42. */
43.
44. int
45. sigma(int m)
46. {
47.     // base case
48.     if (m <= 0)
```

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

```
1. /*****
2.  * string1.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Prints a given string one character per line.
8.  *
9.  * Demonstrates strings as arrays of chars and use of strlen.
10. *****/
11.
12. #include <cs50.h>
13. #include <stdio.h>
14. #include <string.h>
15.
16.
17. int
18. main(void)
19. {
20.     // get line of text
21.     string s = GetString();
22.
23.     // print string, one character per line
24.     if (s != NULL)
25.     {
26.         for (int i = 0; i < strlen(s); i++)
27.         {
28.             char c = s[i];
29.             printf("%c\n", c);
30.         }
31.     }
32. }
```

```
1. /*****
2.  * string2.c
3.  *
4.  * Computer Science 50
5.  * David J. Malan
6.  *
7.  * Prints a given string one character per line.
8.  *
9.  * Demonstrates strings as arrays of chars with slight optimization.
10. *****/
11.
12. #include <cs50.h>
13. #include <stdio.h>
14. #include <string.h>
15.
16.
17. int
18. main(void)
19. {
20.     // get line of text
21.     string s = GetString();
22.
23.     // print string, one character per line
24.     if (s != NULL)
25.     {
26.         for (int i = 0, n = strlen(s); i < n; i++)
27.         {
28.             printf("%c\n", s[i]);
29.         }
30.     }
31. }
```