```
 1: /**************************************************************************
 2:  * fs1.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Computes n'th Fibonacci number.
 8:  *
 9:  * Demonstrates recursion (and bad design).
10:  **************************************************************************/
11:
12: #include <cs50.h>
13: #include <stdio.h>
14: #include <stdlib.h>
15:
16:
17: /* prototype */
18: long long fs(int);
19:
20:
21: int
22: main(int argc, char * argv[])
23: {
24:     int n;
25:
26:     /* ensure proper usage */
27:     if (argc != 2)
28:     {
29:         printf("Usage: %s n\n", argv[0]);
30:         return 1;
31:     }
32:
33:     /* compute and print n'th number in Fibonacci sequence */
34:     n = atoi(argv[1]);
35:     if (n < 0)
36:         printf("Input must be non-negative.\n");
37:     else
38:         printf("fs(%d) = %lld\n", n, fs(n));
39: }
40:
41:
42: /*
43:  * long long
44:  * fs(int n)
45:  *
46:  * Returns n'th number in Fibonacci sequence.
47:  */
48:
49: long long
50: fs(int n)
51: {
52:     /* compute n'th number */
53:     if (n == 0)
54:         return 0;
55:     else if (n == 1)
56:         return 1;
57:     else
58:         return (fs(n-1) + fs(n-2));
59: }
60:
```

```
 1: /****************************************************************************
 2:  * fs2.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Computes n'th Fibonacci number as well as the number of times
 8:  * every number in the sequence is computed.
 9:  *
10:  * Demonstrates bad design (and instrumentation).
11:  ***************************************************************************/
12:
13: #include <cs50.h>
14: #include <stdio.h>
15: #include <stdlib.h>
16:
17:
18: /* prototype */
19: long long fs(int);
20:
21: /* instrumentation */
22: long long * counts;
23:
24:
25: int
26: main(int argc, char * argv[])
27: {
28:     int i, n;
29:
30:     /* ensure proper usage */
31:     if (argc != 2)
32:     {
33:         printf("Usage: %s n\n", argv[0]);
34:         return 1;
35:     }
36:
37:     /* validate n */
38:     n = atoi(argv[1]);
39:     if (n < 0)
40:     {
41:         printf("Input must be non-negative.\n");
42:         return 1;
43:     }
44:
45:     /* allocate space for array of counts */
46:     counts = (long long *) malloc(n * sizeof(long long));
47:     if (counts == NULL)
48:     {
49:         printf("Out of memory!\n");
50:         return 2;
51:     }
52:     for (i = n; i >= 0; i--)
53:         counts[i] = 0;
54:
55:     /* compute and print n'th number in Fibonacci sequence */
56:     printf("fs(%d) = %lld\n", n, fs(n));
57:     for (i = n; i >= 0; i--)
58:         printf("fs(%d) was called %lld time(s)\n", i, counts[i]);
59: }
60:
61:
62: /*
63:  * long long
64:  * fs(int n)
```

```
 *
 * Returns n’th number in Fibonacci sequence.
 */

long long
fs(int n)
{
    /* instrumentation */
    counts[n]++;

    /* compute n’th number */
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return (fs(n-1) + fs(n-2));
}

```

```
 1: /****************************************************************************
 2:  * fs3.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Computes n'th Fibonacci number.
 8:  *
 9:  * Demonstrates dynamic programming (i.e., memoization).
10:  ****************************************************************************/
11:
12: #include <cs50.h>
13: #include <stdio.h>
14: #include <stdlib.h>
15:
16:
17: /* prototype */
18: long long fs(int);
19:
20: /* cache of answers */
21: long long * memo;
22:
23: /* sentinel value indicating absence of a memoized answer */
24: const long long SENTINEL = -1;
25:
26:
27: int
28: main(int argc, char * argv[])
29: {
30:     int i, n;
31:
32:     /* ensure proper usage */
33:     if (argc != 2)
34:     {
35:         printf("Usage: %s n\n", argv[0]);
36:         return 1;
37:     }
38:
39:     /* validate n */
40:     n = atoi(argv[1]);
41:     if (n < 0)
42:     {
43:         printf("Input must be non-negative.\n");
44:         return 1;
45:     }
46:
47:     /* instantiate memo */
48:     memo = (long long *) malloc((n+1) * sizeof(long long));
49:     if (memo == NULL)
50:     {
51:         printf("Out of memory!\n");
52:         return 2;
53:     }
54:
55:     /* initialize memo, using sentinel for answers not yet computed */
56:     memo[0] = 0;
57:     memo[1] = 1;
58:     for (i = 2; i <= n; i++)
59:         memo[i] = SENTINEL;
60:
61:     /* compute and print n'th number in Fibonacci sequence */
62:     printf("fs(%d) = %lld\n", n, fs(n));
63: }
64:
```

```
65:
66: /*
67:  * long long
68:  * fs(int n)
69:  *
70:  * Returns n’th number in Fibonacci sequence.
71:  */
72:
73: long long
74: fs(int n)
75: {
76:     /* compute n’th number */
77:     if (memo[n] != SENTINEL)
78:         return memo[n];
79:     else
80:         return (memo[n] = fs(n-1) + fs(n-2));
81: }
82:
```

```
 1: /**********************************************************************
 2:  * gdb.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Offers opportunities to poke around with GDB.
 8:  **********************************************************************/
 9:
10: #include <stdio.h>
11:
12:
13: int  foo(int n);
14: void bar(int m);
15:
16: int
17: main(int argc, char * argv[])
18: {
19:         int a;
20:         a = 5;
21:         foo(a);
22:         return 0;
23: }
24:
25: int
26: foo(int n)
27: {
28:         int b;
29:         b = n;
30:         b *= 2;
31:         bar(b);
32:         return b;
33: }
34:
35: void
36: bar(int m)
37: {
38:         printf("Hi, I'm bar!\n");
39: }
40:
```