```
1.  /*****************************************************************************
2.   * argv-0.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Prints command-line arguments, one per line.
8.   *
9.   * Demonstrates use of argv.
10.  *****************************************************************************/
11.
12.  #include <cs50.h>
13.  #include <stdio.h>
14.
15.  int main(int argc, string argv[])
16.  {
17.      // print arguments
18.      printf("\n");
19.      for (int i = 0; i < argc; i++)
20.      {
21.          printf("%s\n", argv[i]);
22.      }
23.      printf("\n");
24.  }
```

```
1.  /*****************************************************************************
2.   * argv-1.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Prints command-line arguments, one character per line.
8.   *
9.   * Demonstrates argv as a two-dimensional array.
10.  *****************************************************************************/
11.
12.  #include <cs50.h>
13.  #include <stdio.h>
14.  #include <string.h>
15.
16.  int main(int argc, string argv[])
17.  {
18.      // print arguments
19.      printf("\n");
20.      for (int i = 0; i < argc; i++)
21.      {
22.          for (int j = 0, n = strlen(argv[i]); j < n; j++)
23.          {
24.              printf("%c\n", argv[i][j]);
25.          }
26.          printf("\n");
27.      }
28.  }
```

```c
1.  /******************************************************************************
2.   * array.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
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.  // number of quizzes per term
17.  #define QUIZZES 2
18.
19.  int main(void)
20.  {
21.      // ask user for grades
22.      float grades[QUIZZES];
23.      printf("\nWhat were your quiz scores?\n\n");
24.      for (int i = 0; i < QUIZZES; i++)
25.      {
26.          printf("Quiz #%i of %i: ", i + 1, QUIZZES);
27.          grades[i] = GetFloat();
28.      }
29.
30.      // compute average
31.      float sum = 0;
32.      for (int i = 0; i < QUIZZES; i++)
33.      {
34.          sum += grades[i];
35.      }
36.      int average = (int) round(sum / QUIZZES);
37.
38.      // report average
39.      printf("\nYour average is: %i\n\n", average);
40.  }
```

```
1.   /**
2.    * ascii-0.c
3.    *
4.    * David J. Malan
5.    * malan@harvard.edu
6.    *
7.    * Displays the mapping between alphabetical ASCII characters and
8.    * their decimal equivalents.
9.    *
10.   * Demonstrates casting from int to char.
11.   */
12.
13.  #include <stdio.h>
14.
15.  int main(void)
16.  {
17.      // display mapping for uppercase letters
18.      for (int i = 65; i < 65 + 26; i++)
19.      {
20.          printf("%c: %i\n", (char) i, i);
21.      }
22.
23.      // separate uppercase from lowercase
24.      printf("\n");
25.
26.      // display mapping for lowercase letters
27.      for (int i = 97; i < 97 + 26; i++)
28.      {
29.          printf("%c: %i\n", (char) i, i);
30.      }
31.  }
```

```
1.  /**
2.   * ascii-1.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Displays the mapping between alphabetical ASCII characters and
8.   * their decimal equivalents.
9.   *
10.  * Demonstrates casting from char to int.
11.  */
12.
13.  #include <stdio.h>
14.
15.  int main(void)
16.  {
17.      // display mapping for uppercase letters
18.      for (int c = 'A'; c <= 'Z'; c++)
19.      {
20.          printf("%c: %i\n", c, (int) c);
21.      }
22.
23.      // separate uppercase from lowercase
24.      printf("\n");
25.
26.      // display mapping for lowercase letters
27.      for (int c = 'a'; c <= 'z'; c++)
28.      {
29.          printf("%c: %i\n", i, (int) i);
30.      }
31.  }
```

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

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

```
1.   /******************************************************************************
2.    * beer-2.c
3.    *
4.    * David J. Malan
5.    * malan@harvard.edu
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.  int main(void)
16.  {
17.      // ask user for number
18.      printf("How many bottles will there be? ");
19.      int n = GetInt();
20.
21.      // exit upon invalid input
22.      if (n < 1)
23.      {
24.          printf("Sorry, that makes no sense.\n");
25.          return 1;
26.      }
27.
28.      // sing the annoying song
29.      printf("\n");
30.      for (int i = n; i > 0; i--)
31.      {
32.          string s = (i == 1) ? "bottle" : "bottles";
33.          string t = (i == 2) ? "bottle" : "bottles";
34.
35.          // sing verses
36.          printf("%i %s of beer on the wall,\n", i, s);
37.          printf("%i %s of beer,\n", i, s);
38.          printf("Take one down, pass it around,\n");
39.          printf("%i %s of beer on the wall.\n\n", i - 1, t);
40.      }
41.  }
```

```
1.  /*****************************************************************************
2.   * beer-3.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
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.  // function prototype
16.  void chorus(int b);
17.
18.  int 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.          chorus(i);
36.      }
37.  }
38.
39.  /**
40.   * Sings about specified number of bottles.
41.   */
42.  void chorus(int b)
43.  {
44.      // use proper grammar
45.      string s = (b == 1) ? "bottle" : "bottles";
46.      string t = (b == 2) ? "bottle" : "bottles";
47.
48.      // sing verses
```

```
49.        printf("%i %s of beer on the wall,\n", b, s);
50.        printf("%i %s of beer,\n", b, s);
51.        printf("Take one down, pass it around,\n");
52.        printf("%i %s of beer on the wall.\n\n", b - 1, t);
53.  }
```

```c
 1.  /****************************************************************************
 2.   * buggy-0.c
 3.   *
 4.   * David J. Malan
 5.   * malan@harvard.edu
 6.   *
 7.   * Should print 10 asterisks but doesn't!
 8.   * Can you find the bug?
 9.   ****************************************************************************/
10.
11.  #include <stdio.h>
12.
13.  int main(void)
14.  {
15.      for (int i = 0; i <= 10; i++)
16.          printf("*");
17.  }
```

```
 1.  /*****************************************************************************
 2.   * buggy-1.c
 3.   *
 4.   * David J. Malan
 5.   * malan@harvard.edu
 6.   *
 7.   * Should print 10 asterisks, one per line, but doesn't!
 8.   * Can you find the bug?
 9.   *****************************************************************************/
10.
11.  #include <stdio.h>
12.
13.  int main(void)
14.  {
15.      for (int i = 0; i <= 10; i++)
16.          printf("*");
17.          printf("\n");
18.  }
```

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

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

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

```
1.  /*****************************************************************************
2.   * capitalize.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Capitalizes a given string.
8.   *
9.   * Demonstrates casting and iteration over strings as arrays of chars.
10.  *****************************************************************************/
11.
12.  #include <cs50.h>
13.  #include <stdio.h>
14.  #include <string.h>
15.
16.  int main(void)
17.  {
18.      // get line of text
19.      string s = GetString();
20.
21.      // capitalize text
22.      for (int i = 0, n = strlen(s); i < n; i++)
23.      {
24.          if (s[i] >= 'a' && s[i] <= 'z')
25.          {
26.              printf("%c", s[i] - ('a' - 'A'));
27.          }
28.          else
29.          {
30.              printf("%c", s[i]);
31.          }
32.      }
33.      printf("\n");
34.  }
```

```
1.  /******************************************************************************
2.   * f2c.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Converts Fahrenheit to Celsius.
8.   *
9.   * Demonstrates arithmetic.
10.   ******************************************************************************/
11.
12.  #include <cs50.h>
13.  #include <stdio.h>
14.
15.  int main(void)
16.  {
17.      // ask user user for temperature in Fahrenheit
18.      printf("Temperature in F: ");
19.      float f = GetFloat();
20.
21.      // convert F to C
22.      float c = 5.0 / 9.0 * (f - 32.0);
23.
24.      // display result to one decimal place
25.      printf("%.1f\n", c);
26.  }
```

```
1.  /**
2.   * floats-0.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Tries to print 1/10 as a floating-point value.
8.   *
9.   * Demonstrates truncation.
10.  */
11.
12.  #include <stdio.h>
13.
14.  int main(void)
15.  {
16.      float f = 1 / 10;
17.      printf("%.1f\n", f);
18.  }
```

```
1.  /**
2.   * floats-1.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Prints 1/10 as a floating-point value to one decimal place.
8.   *
9.   * Demonstrates division of floating-point values.
10.  */
11.
12.  #include <stdio.h>
13.
14.  int main(void)
15.  {
16.      float f = 1.0 / 10.0;
17.      printf("%.1f\n", f);
18.  }
```

```
1.  /**
2.   * floats-2.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Prints 1/10 as a floating-point value to 28 decimal places.
8.   *
9.   * Demonstrates imprecision of floating-point values.
10.  */
11.
12.  #include <stdio.h>
13.
14.  int main(void)
15.  {
16.      float f = 1.0 / 10.0;
17.      printf("%.28f\n", f);
18.  }
```

```c
 1.  /******************************************************************************
 2.   * global.c
 3.   *
 4.   * David J. Malan
 5.   * malan@harvard.edu
 6.   *
 7.   * Increments variables.
 8.   *
 9.   * Demonstrates use of global variable and issue of scope.
10.   ******************************************************************************/
11.
12.  #include <stdio.h>
13.
14.  // global variable
15.  int x;
16.
17.  // function prototype
18.  void increment(void);
19.
20.  int main(void)
21.  {
22.      printf("x is now %i\n", x);
23.      printf("Initializing...\n");
24.      x = 1;
25.      printf("Initialized!\n");
26.      printf("x is now %i\n", x);
27.      printf("Incrementing...\n");
28.      increment();
29.      printf("Incremented!\n");
30.      printf("x is now %i\n", x);
31.  }
32.
33.  /**
34.   * Increments x.
35.   */
36.  void increment(void)
37.  {
38.      x++;
39.  }
```

```c
 1.  /* http://www.ioccc.org/years.html */
 2.
 3.  #include "stdio.h"
 4.  #define e 3
 5.  #define g (e/e)
 6.  #define h ((g+e)/2)
 7.  #define f (e-g-h)
 8.  #define j (e*e-g)
 9.  #define k (j-h)
10.  #define l(x) tab2[x]/h
11.  #define m(n,a) ((n&(a))==(a))
12.
13.  long tab1[]={ 989L,5L,26L,0L,88319L,123L,0L,9367L };
14.  int tab2[]={ 4,6,10,14,22,26,34,38,46,58,62,74,82,86 };
15.
16.  main(m1,s) char *s; {
17.      int a,b,c,d,o[k],n=(int)s;
18.      if(m1==1){ char b[2*j+f-g]; main(l(h+e)+h+e,b); printf(b); }
19.      else switch(m1-=h){
20.      case f:
21.          a=(b=(c=(d=g)<<g)<<g)<<g;
22.          return(m(n,a|c)|m(n,b)|m(n,a|d)|m(n,c|d));
23.      case h:
24.          for(a=f;a<j;++a)if(tab1[a]&&!(tab1[a]%((long)l(n))))return(a);
25.      case g:
26.          if(n<h)return(g);
27.          if(n<j){n-=g;c='D';o[f]=h;o[g]=f;}
28.          else{c='\r'-'\b';n-=j-g;o[f]=o[g]=g;}
29.          if((b=n)>=e)for(b=g<<g;b<n;++b)o[b]=o[b-h]+o[b-g]+c;
30.          return(o[b-g]%n+k-h);
31.      default:
32.          if(m1-=e) main(m1-g+e+h,s+g); else *(s+g)=f;
33.          for(*s=a=f;a<e;) *s=(*s<<e)|main(h+a++,(char *)m1);
34.      }
35.  }
36.
```

```
 1.  /*
 2.
 3.     iUnlock v42.PROPER -- Copyright 2007 The dev team
 4.
 5.     Credits: Daeken, Darkmen, guest184, gray, iZsh, pytey, roxfan, Sam, uns, Zappaz, Zf
 6.
 7.     All code, information or data [from now on "data"] available
 8.     from the "iPhone dev team" [1] or any other project linked from
 9.     this or other pages is owned by the creator who created the data.
10.     The copyright, license right, distribution right and any other
11.     rights lies with the creator.
12.
13.     It is prohibitied to use the data without the written agreement
14.     of the creator. This included using ideas in other projects
15.     (commercial or not commercial).
16.
17.     Where data was created by more than 1 creator a written agreement
18.     from each of the creators has to be obtained.
19.
20.     Punishment: Monkeys coming out of your ass Bruce Almighty style.
21.
22.     [1] http://iphone.fiveforty.net/wiki/index.php?title=Main_Page
23.  */
24.  #include <stdio.h>
25.  #include <stdlib.h>
26.  #include <unistd.h>
27.  #include <string.h>
28.  #include <fcntl.h>
29.  #include <termios.h>
30.  #include <errno.h>
31.  #include <time.h>
32.  #include "IOKit/IOKitLib.h"
33.
34.  #include "packets.h"
35.
36.  #define ever ;;
37.  #define LOG stdout
38.  #define SPEED 750000
39.
40.  #pragma pack(1)
41.
42.  #define BUFSIZE (65536+100)
43.  unsigned char readbuf[BUFSIZE];
44.
45.  struct termios term;
46.
47.  //#define DEBUG_ENABLED 1
48.
```

```
49.  #ifndef DEBUG_ENABLED
50.  #define DEBUGLOG(x)
51.  #else
52.  #define DEBUGLOG(x) x
53.  #endif
54.
55.  #define UINT(x) *((unsigned int *) (x))
56.
57.  const char * RE = "Why the hell are you reversing this app?! We said we were "\
58.  "going to release the sources...";
59.
60.  void HexDumpLine(unsigned char *buf, int remainder, int offset)
61.  {
62.    int i = 0;
63.    char c = 0;
64.
65.    // Print the hex part
66.    fprintf(LOG, "%08x | ", offset);
67.    for (i = 0; i < 16; ++i) {
68.      if (i < remainder)
69.        fprintf(LOG, "%02x%s", buf[i], (i == 7) ? "  " : " ");
70.      else
71.        fprintf(LOG, "  %s", (i == 7) ? "  " : " ");
72.    }
73.    // Print the ascii part
74.    fprintf(LOG, " | ");
75.    for (i = 0; i < 16 && i < remainder; ++i) {
76.      c = buf[i];
77.      if (c >= 0x20 && c <= 0x7e)
78.        fprintf(LOG, "%c%s", c, (i == 7) ? " " : "");
79.      else
80.        fprintf(LOG, ".%s", (i == 7) ? " " : "");
81.    }
82.
83.    fprintf(LOG, "\n");
84.  }
85.
86.  void HexDump(unsigned char *buf, int size)
87.  {
88.    int i = 0;
89.
90.    for (i = 0; i < size; i += 16)
91.      HexDumpLine(buf + i, size - i, i);
92.    fprintf(LOG, "%08x\n", size);
93.  }
94.
95.  int Checksum(CmdHeader * packet)
96.  {
```

```
97.     int sum = 0x00030000;
98.     sum += packet->opcode;
99.     sum += packet->param_len;
100.
101.    int len = packet->param_len;
102.    unsigned char * buf = ((unsigned char *)packet) + sizeof (CmdHeader);
103.    int i = 0;
104.
105.    for (i = 0; i < len; ++i)
106.      sum += buf[i];
107.    return sum;
108. }
109.
110. void SendCmd(int fd, void *buf, size_t size)
111. {
112.    DEBUGLOG(fprintf(LOG, "Sending:\n"));
113.    DEBUGLOG(HexDump((unsigned char*)buf, size));
114.
115.    if(write(fd, buf, size) == -1) {
116.      fprintf(stderr, "Shit. %s\n", strerror(errno));
117.      exit(1);
118.    }
119. }
120.
121. #define sendBytes(fd, args...) {\
122.    unsigned char sendbuf[] = {args}; \
123.    SendCmd(fd, sendbuf, sizeof(sendbuf)); \
124. }
125.
126. int ReadResp(int fd)
127. {
128.    int len = 0;
129.    struct timeval timeout;
130.    int nfds = fd + 1;
131.    fd_set readfds;
132.
133.    FD_ZERO(&readfds);
134.    FD_SET(fd, &readfds);
135.
136.    // Wait a second
137.    timeout.tv_sec = 0;
138.    timeout.tv_usec = 500000;
139.
140.    while (select(nfds, &readfds, NULL, NULL, &timeout) > 0)
141.      len += read(fd, readbuf + len, BUFSIZE - len);
142.
143.    if (len > 0) {
144.      DEBUGLOG(fprintf(LOG, "Read:\n"));
```

```
145.         DEBUGLOG(HexDump(readbuf, len));
146.     }
147.     return len;
148. }
149.
150. int InitConn(int speed)
151. {
152.     int fd = open("/dev/tty.baseband", O_RDWR | 0x20000 | O_NOCTTY);
153.     unsigned int blahnull = 0;
154.     unsigned int handshake = TIOCM_DTR | TIOCM_RTS | TIOCM_CTS | TIOCM_DSR;
155.
156.     if(fd == -1) {
157.         fprintf(stderr, "%i(%s)\n", errno, strerror(errno));
158.         exit(1);
159.     }
160.
161.     ioctl(fd, 0x2000740D);
162.     fcntl(fd, 4, 0);
163.     tcgetattr(fd, &term);
164.
165.     ioctl(fd, 0x8004540A, &blahnull);
166.     cfsetspeed(&term, speed);
167.     cfmakeraw(&term);
168.     term.c_cc[VMIN] = 0;
169.     term.c_cc[VTIME] = 5;
170.
171.     term.c_iflag = (term.c_iflag & 0xFFFFF0CD) | 5;
172.     term.c_oflag =  term.c_oflag & 0xFFFFFFFE;
173.     term.c_cflag = (term.c_cflag & 0xFFFC6CFF) | 0x3CB00;
174.     term.c_lflag =  term.c_lflag & 0xFFFFFA77;
175.
176.     term.c_cflag = (term.c_cflag & ~CSIZE) | CS8;
177.     term.c_cflag &= ~PARENB;
178.     term.c_lflag &= ~ECHO;
179.
180.     tcsetattr(fd, TCSANOW, &term);
181.
182.     ioctl(fd, TIOCSDTR);
183.     ioctl(fd, TIOCCDTR);
184.     ioctl(fd, TIOCMSET, &handshake);
185.
186.     return fd;
187. }
188.
189. void RestartBaseband()
190. {
191.     kern_return_t    result;
192.     mach_port_t      masterPort;
```

```
193.
194.    result = IOMasterPort(MACH_PORT_NULL, &masterPort);
195.    if (result) {
196.      DEBUGLOG(printf("IOMasterPort failed\n"));
197.      return;
198.    }
199.
200.    CFMutableDictionaryRef matchingDict = IOServiceMatching("AppleBaseband");
201.    io_service_t service = IOServiceGetMatchingService(kIOMasterPortDefault, matchingDict);
202.    if (!service) {
203.      DEBUGLOG(printf("IOServiceGetMatchingService failed\n"));
204.      return;
205.    }
206.
207.    io_connect_t conn;
208.    result = IOServiceOpen(service, mach_task_self(), 0, &conn);
209.    if (result) {
210.      DEBUGLOG(printf("IOServiceOpen failed\n"));
211.      return;
212.    }
213.
214.    result = IOConnectCallScalarMethod(conn, 0, 0, 0, 0, 0);
215.    if (result == 0)
216.      DEBUGLOG(printf("Baseband reset.\n"));
217.    else
218.      DEBUGLOG(printf("Baseband reset failed\n"));
219.    IOServiceClose(conn);
220. }
221.
222. void SendGetVersion(int fd)
223. {
224.    sendBytes(fd, 0x60, 0x0D);
225. }
226.
227. void GetVersion(int fd)
228. {
229.    SendGetVersion(fd);
230.
231.    for (ever) {
232.      if(ReadResp(fd) != 0) {
233.        if(readbuf[0] == 0x0b)
234.          break;
235.      }
236.      SendGetVersion(fd);
237.    }
238.
239.    VersionAck *ver = (VersionAck *) readbuf;
240.    DEBUGLOG(printf("Boot mode: %02X\n", ver->bootmode));
```

```
241.     DEBUGLOG(printf("Major: %d, Minor: %d\n", ver->major, ver->minor));
242.     DEBUGLOG(printf("Version: %s\n", ver->version));
243. }
244.
245. void CFIStage1_2(int fd)
246. {
247.    CFIStage1Req req;
248.    req.cmd.cls = 0x2;
249.    req.cmd.opcode = BBCFISTAGE1;
250.    req.cmd.param_len = 0;
251.    req.checksum = Checksum ((CmdHeader*)&req);
252.    DEBUGLOG(printf("Sending CFIStage1 Request\n"));
253.    SendCmd(fd, &req, sizeof (CFIStage1Req));
254.    DEBUGLOG(printf("Receiving CFIStage1 response\n"));
255.    if (!ReadResp(fd)) {
256.       DEBUGLOG(fprintf(stderr, "Failed to receive CFIStage1 response\n"));
257.       exit(1);
258.    }
259.    CFIStage1Ack * cfi1resp = (CFIStage1Ack *) readbuf;
260.    cfi1resp->cmd.opcode = BBCFISTAGE2;
261.    cfi1resp->checksum = Checksum((CmdHeader*)cfi1resp);
262.    SendCmd(fd, cfi1resp, sizeof(CFIStage1Ack));
263.    if (!ReadResp(fd)) {
264.       DEBUGLOG(fprintf(stderr, "Failed to receive CFIStage2 response\n"));
265.       exit(1);
266.    }
267. }
268.
269. void ReadSecpack(const char * FilePath, void * Buffer)
270. {
271.    FILE * fp = fopen(FilePath, "rb");
272.    if (fp == NULL) {
273.       perror(FilePath);
274.       exit(1);
275.    }
276.
277.    fseek(fp, 0x1a4L, SEEK_SET);
278.    if (fread(Buffer, 1, 0x800, fp) != 0x800) {
279.       fprintf(stderr, "Error while reading the secpack content\n");
280.       free(Buffer);
281.       exit(1);
282.    }
283.    fclose(fp);
284. }
285.
286. void SendBeginSecpack(int fd, void * Secpack)
287. {
288.    printf("Sending Begin Secpack command\n");
```

```
289.
290.    BeginSecpackReq req;
291.    req.cmd.cls = 0x2;
292.    req.cmd.opcode = BBBEGINSECPACK;
293.    req.cmd.param_len = 0x800;
294.    memcpy(&req.data, Secpack, 0x800);
295.    req.checksum = Checksum((CmdHeader*)&req);
296.    SendCmd(fd, &req, sizeof (BeginSecpackReq));
297.    // Wait for the answer
298.    DEBUGLOG(printf("Reading answer\n"));
299.    while (!ReadResp(fd)) ;
300. }
301.
302. void SendEndSecpack(int fd)
303. {
304.    printf("Sending End Secpack command\n");
305.
306.    EndSecpackReq req;
307.    req.cmd.cls = 0x2;
308.    req.cmd.opcode = BBENDSECPACK;
309.    req.cmd.param_len = 0x2;
310.    req.unknown = 0;
311.    req.checksum = Checksum((CmdHeader*)&req);
312.    SendCmd(fd, &req, sizeof (EndSecpackReq));
313.    DEBUGLOG(printf("Reading answer\n"));
314.    ReadResp(fd);
315. }
316.
317. void SendErase(int fd, int BeginAddr, int EndAddr)
318. {
319.    printf("Sending Erase command\n");
320.
321.    EraseReq req;
322.    req.cmd.cls = 0x2;
323.    req.cmd.opcode = BBERASE;
324.    req.cmd.param_len = 8;
325.    req.low_addr = BeginAddr;
326.    req.high_addr = EndAddr;
327.    req.checksum = Checksum((CmdHeader*)&req);
328.    SendCmd(fd, &req, sizeof (EraseReq));
329.    sleep(1); // Give it some time
330.    DEBUGLOG(printf("Reading answer\n"));
331.    if (!ReadResp(fd)) {
332.      fprintf(stderr, "Ooops, something was wrong while erasing\n");
333.      exit(1);
334.    }
335.
336.    printf("Waiting For Erase Completion...\n");
```

```
337.
338.     EraseAck * eraseack = (EraseAck *) readbuf;
339.     EraseStatusReq statusreq;
340.     statusreq.cmd.cls = 0x2;
341.     statusreq.cmd.opcode = BBERASESTATUS;
342.     statusreq.cmd.param_len = 2;
343.     statusreq.unknown1 = eraseack->unknown1;
344.     statusreq.checksum = Checksum((CmdHeader*)&statusreq);
345.
346.     EraseStatusAck * erasestatusack = (EraseStatusAck *) readbuf;
347.     do {
348.         SendCmd(fd, &statusreq, sizeof(EraseStatusReq));
349.         DEBUGLOG(printf("Reading answer\n"));
350.         ReadResp(fd);
351.         erasestatusack = (EraseStatusAck *) readbuf;
352.     } while (erasestatusack->done != 1);
353.
354. }
355.
356. int ReadAddr(int fd, unsigned short int size)
357. {
358.     ReadReq req;
359.
360.     req.cmd.cls = 0x2;
361.     req.cmd.opcode = BBREAD;
362.     req.cmd.param_len = 0x2;
363.     req.size = size;
364.     req.checksum = Checksum((CmdHeader*)&req);
365.
366.     DEBUGLOG(printf("\nSending read request:\n"));
367.     SendCmd(fd, &req, sizeof(ReadReq));
368.
369.     DEBUGLOG(printf("Receiving read response\n"));
370.     return ReadResp(fd);
371. }
372.
373. void Seek(int fd, unsigned int addr)
374. {
375.     DEBUGLOG(printf("Sending seek command for addr %p\n", addr));
376.     SeekReq req;
377.     req.cmd.cls = 0x2;
378.     req.cmd.opcode = BBSEEK;
379.     req.cmd.param_len = 0x4;
380.     req.addr = addr;
381.     req.checksum = Checksum((CmdHeader*)&req);
382.     SendCmd(fd, &req, sizeof (SeekReq));
383.     DEBUGLOG(printf("Reading answer\n"));
384.     ReadResp(fd);
```

```
385.  }
386.
387.  void DumpReadBufToFile(FILE * fp)
388.  {
389.    ReadAck * packet = (ReadAck*)readbuf;
390.    int len = packet->cmd.param_len;
391.    unsigned char * buf = &packet->first_char;
392.    fwrite(buf, len, 1, fp);
393.  }
394.
395.  void Dump(int fd, FILE * fp)
396.  {
397.    unsigned int addr = 0xa0000000;
398.    unsigned int nor_size = 0x400000; // the NOR is 4M (32Mbit)
399.    unsigned int page_size = 0x800;
400.    int i = 0;
401.
402.    Seek(fd, addr);
403.    for (i = 0; i < nor_size; i += page_size) {
404.      DEBUGLOG(printf("Addr: %p\n", addr + i));
405.      ReadAddr(fd, page_size);
406.      DumpReadBufToFile(fp);
407.    }
408.  }
409.
410.  void * ReadBL(const char * FilePath, int * Size)
411.  {
412.    FILE * fp = fopen(FilePath, "rb");
413.    if (fp == NULL) {
414.      perror(FilePath);
415.      exit(1);
416.    }
417.
418.    fseek(fp, 0, SEEK_END);
419.    int size = ftell(fp);
420.    fseek(fp, 0, SEEK_SET);
421.
422.    void * buffer = malloc(size);
423.
424.    if (fread(buffer, 1, size, fp) != size) {
425.      fprintf(stderr, "Error while reading the BL content\n");
426.      free(buffer);
427.      exit(1);
428.    }
429.    fclose(fp);
430.    *Size = size;
431.    return buffer;
432.  }
```

```
433.
434.  void * ReadFW(const char * FilePath, int Size)
435.  {
436.    FILE * fp = fopen(FilePath, "rb");
437.    if (fp == NULL) {
438.      perror(FilePath);
439.      exit(1);
440.    }
441.
442.    void * buffer = malloc(Size);
443.    fseek(fp, 0x9a4L + 0x20000, SEEK_SET);
444.
445.    if (fread(buffer, 1, Size, fp) != Size) {
446.      fprintf(stderr, "Error while reading the FW content\n");
447.      free(buffer);
448.      exit(1);
449.    }
450.    fclose(fp);
451.    return buffer;
452.  }
453.
454.
455.  void SendWriteOnePage(int fd, unsigned char * Buffer, int Size)
456.  {
457.    int size_to_write = Size > 0x800 ? 0x800 : Size;
458.
459.    // Header, buffer, checksum
460.    int req_size = sizeof (CmdHeader) + size_to_write + 4;
461.    WriteReq * req = malloc(req_size);
462.
463.    req->cmd.cls = 0x2;
464.    req->cmd.opcode = BBWRITE;
465.    req->cmd.param_len = size_to_write;
466.    memset(&req->first_char, 0, size_to_write);
467.    memcpy(&req->first_char, Buffer, size_to_write);
468.    *(unsigned int *)(&req->first_char + size_to_write) = Checksum((CmdHeader*)req);
469.
470.    SendCmd(fd, req, req_size);
471.    DEBUGLOG(printf("Reading answer\n"));
472.    if (!ReadResp(fd)) {
473.      free(req);
474.      fprintf(stderr, "Ooops, something was wrong while Writing\n");
475.      exit(1);
476.    }
477.    free(req);
478.  }
479.
480.  void SendWrite(int fd, unsigned char * Buffer, int Size, int Debug)
```

```
481.  {
482.    if (Debug) printf("Sending Write command\n");
483.    int cur_size = Size;
484.    int step = Size / 20;
485.    int last_progress = 0;
486.
487.    while (cur_size > 0) {
488.      int progress = (Size - cur_size) / step;
489.      if (Debug && progress >= last_progress) {
490.        printf("%.2d%%\n", progress * 5);
491.        last_progress = progress;
492.      }
493.      SendWriteOnePage(fd, Buffer, cur_size);
494.      cur_size -= 0x800;
495.      Buffer += 0x800;
496.    }
497.  }
498.
499.  // In progress ;)
500.  void PatchingFW(unsigned char * Buffer)
501.  {
502.    printf("Patching FW\n");
503.
504.    if (Buffer[213740] != 0x04
505.      || Buffer[213741] != 0x00
506.      || Buffer[213742] != 0xa0
507.      || Buffer[213743] != 0xe1)
508.    {
509.      printf("Error in patch\n");
510.      exit(1);
511.    }
512.    Buffer[213740] = 0x00;
513.    Buffer[213741] = 0x00;
514.    Buffer[213742] = 0xa0;
515.    Buffer[213743] = 0xe3;
516.
517.    memset(Buffer + 0x410, 0, 3);
518.    memset(Buffer + 0x800, 0, 16 * 10);
519.    memset(Buffer + 0xBFC, 0, 16 * 8);
520.    memset(Buffer + 0xFFC, 0, 16 * 8);
521.  }
522.
523.  void ValidateFW(int fd, unsigned char * Buffer)
524.  {
525.    printf("Validating the write command\n");
526.    Seek(fd, 0xA0020000);
527.    ReadAddr(fd, 0x800);
528.
```

```c
529.    ReadAck * packet = (ReadAck*)readbuf;
530.    unsigned char * buf = &packet->first_char;
531.
532.    if (memcmp(buf, Buffer, 0x800)) {
533.      printf("FW differences found\n");
534.    } else {
535.      printf("FW are equal!\n");
536.    }
537.  }
538.
539. //void usage(char *prog)
540. //{
541. //  fprintf(stderr, "Usage: %s <fls file> [bl]\n", prog);
542. //  exit(1);
543. //}
544.
545. void usage(char *prog)
546. {
547.   fprintf(stderr, "Usage: %s <fls file> <NOR file>\n", prog);
548.   exit(1);
549. }
550.
551.
552. void credit(void)
553. {
554.   printf("iUnlock v42.PROPER -- Copyright 2007 The dev team\n\n\n"\
555.          "Credits: Daeken, Darkmen, guest184, gray, iZsh, pytey, roxfan, Sam, uns, Zappaz, Zf\n\n" \
556.          "* Leet Hax not for commercial uses\n"\
557.          "  Punishment: Monkeys coming out of your ass Bruce Almighty style.\n\n"
558.          );
559. }
560.
561. int main(int argc, char **argv)
562. {
563.   const char * hehe = RE;
564.   int fd;
565.
566.   credit();
567.
568.   if (argc != 3)
569.     usage(argv[0]);
570.
571.   void * secpack = malloc(0x800);
572.   ReadSecpack(argv[1], secpack);
573.
574.   void * fw = NULL;
575.   int fwsize = 0;
576.   fw = ReadBL(argv[2], &fwsize);
```

```
577.
578.    RestartBaseband();
579.    fd = InitConn(115200);
580.
581.    GetVersion(fd);
582.    CFIStage1_2(fd);
583.    SendBeginSecpack(fd, secpack);
584.    SendErase(fd, 0xA0020000, 0xA03bfffe);
585.    Seek(fd, 0xA0020000 - 0x400);
586.    unsigned char foo[0x400];
587.    memset(foo, 0, 0x400);
588.    SendWrite(fd, foo, 0x400, false);
589.    SendWrite(fd, fw, fwsize, true);
590.    SendEndSecpack(fd);
591.    ValidateFW(fd, fw);
592.    printf("Completed.\nEnjoy!\n");
593.    free(fw);
594.
595.    return 0;
596. }
```

```c
 1.  /*****************************************************************************
 2.   * nonswitch.c
 3.   *
 4.   * David J. Malan
 5.   * malan@harvard.edu
 6.   *
 7.   * Assesses the size of user's input.
 8.   *
 9.   * Demonstrates use of Boolean ANDing.
10.   *****************************************************************************/
11.
12.  #include <cs50.h>
13.  #include <stdio.h>
14.
15.  int main(void)
16.  {
17.      // ask user for an integer
18.      printf("Give me an integer between 1 and 10: ");
19.      int n = GetInt();
20.
21.      // judge user's input
22.      if (n >= 1 && n <= 3)
23.      {
24.          printf("You picked a small number.\n");
25.      }
26.      else if (n >= 4 && n <= 6)
27.      {
28.          printf("You picked a medium number.\n");
29.      }
30.      else if (n >= 7 && n <= 10)
31.      {
32.          printf("You picked a big number.\n");
33.      }
34.      else
35.      {
36.          printf("You picked an invalid number.\n");
37.      }
38.  }
```

```
1.  /**
2.   * positive.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Demands that user provide a positive integer.
8.   *
9.   * Demonstrates use of do-while.
10.  */
11.
12.  #include <cs50.h>
13.  #include <stdio.h>
14.
15.  int main(void)
16.  {
17.      // loop until user provides a positive integer
18.      int n;
19.      do
20.      {
21.          printf("Please give me a positive int: ");
22.          n = GetInt();
23.      }
24.      while (n < 1);
25.      printf("Thanks for the positive int!\n", n);
26.  }
```

```
1.  /******************************************************************************
2.   * return.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
6.   *
7.   * Cubes a variable.
8.   *
9.   * Demonstrates use of parameter and return value.
10.  ******************************************************************************/
11.
12. #include <stdio.h>
13.
14. // function prototype
15. int cube(int a);
16.
17. int main(void)
18. {
19.     int x = 2;
20.     printf("x is now %i\n", x);
21.     printf("Cubing...\n");
22.     x = cube(x);
23.     printf("Cubed!\n");
24.     printf("x is now %i\n", x);
25. }
26.
27. /**
28.  * Cubes argument.
29.  */
30. int cube(int n)
31. {
32.     return n * n * n;
33. }
```

```
 1.  /**
 2.   * sizeof.c
 3.   *
 4.   * David J. Malan
 5.   * malan@harvard.edu
 6.   *
 7.   * Reports the sizes of C's data types.
 8.   *
 9.   * Demonstrates use of sizeof.
10.   */
11.
12.  #include <stdio.h>
13.
14.  int main(void)
15.  {
16.      // some sample variables
17.      char c;
18.      double d;
19.      float f;
20.      int i;
21.
22.      // report the sizes of variables' types
23.      printf("char: %i\n", sizeof(c));
24.      printf("double: %i\n", sizeof(d));
25.      printf("float: %i\n", sizeof(f));
26.      printf("int: %i\n", sizeof(i));
27.  }
```

```
1.  /**
2.   * string.c
3.   *
4.   * David J. Malan
5.   * malan@harvard.edu
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.  int main(void)
17.  {
18.      // get line of text
19.      string s = GetString();
20.
21.      // print string, one character per line
22.      if (s != NULL)
23.      {
24.          for (int i = 0, n = strlen(s); i < n; i++)
25.          {
26.              printf("%c\n", s[i]);
27.          }
28.      }
29.  }
```

```c
 1.  /*****************************************************************************
 2.   * switch.c
 3.   *
 4.   * David J. Malan
 5.   * malan@harvard.edu
 6.   *
 7.   * Assesses the size of user's input.
 8.   *
 9.   * Demonstrates use of a switch.
10.   *****************************************************************************/
11.
12.  #include <cs50.h>
13.  #include <stdio.h>
14.
15.  int main(void)
16.  {
17.      // ask user for an integer
18.      printf("Give me an integer between 1 and 10: ");
19.      int n = GetInt();
20.
21.      // judge user's input
22.      switch (n)
23.      {
24.          case 1:
25.          case 2:
26.          case 3:
27.              printf("You picked a small number.\n");
28.              break;
29.
30.          case 4:
31.          case 5:
32.          case 6:
33.              printf("You picked a medium number.\n");
34.              break;
35.
36.          case 7:
37.          case 8:
38.          case 9:
39.          case 10:
40.              printf("You picked a big number.\n");
41.              break;
42.
43.          default:
44.              printf("You picked an invalid number.\n");
45.      }
46.  }
```

```
1.  /* http://www.ioccc.org/years.html */
2.
3.                     int
4.                 X=320        ,Y=200,
5.              n=0,m,      x,y,    j=1024;
6.            double     T=44.0      /7,P[
7.            333333      ],C[5]         ={ 0,3,
8.            0,0,8}          ,p=1,            B=11.0
9.            /630,        f=0,r    =        3,g
10.           =7,b          =13,*q=P,    D,*J;
11.            unsigned                  char
12.             U[66666],*v=U,*h,l[5555]
13.                  ,c=0,*e,*a,*z;
14.
15.              #include <math.h>
16.            #define R1(t)    t=(int)(t\
17.          *123456789              )%j; t/=j;
18.           #define            Rl(C,t)\
19.           n++[C]         =         t*n/12;
20.           #define     RI(C)     B=-B; R1\
21.           (r)R1(g      )R1(b      )for(n\
22.            =0; n<j; ){ Rl(C    ,r)Rl\
23.               (C,g)Rl(C      ,b)++n; }
24.
25.
26.
27.        #ifdef __DJGPP__
28.         #include <sys/movedata.h>
29.             #include <dpmi.h>
30.               #include <pc.h>
31.    #define          Q(u,v)            u##portb(0x3##v
32.      #define          W          ; Q(out,C9),*h++/4)
33.          void        F(int i){  __dpmi_regs r
34.         ; if(i){ for(; i>=0; i-=8)while(
35.                  ~Q(in,DA)
36.               )&8^i); for(m=0,z
37.          =h+j; h    <z; m      ++){ Q(
38.        out,C8),m      )W W W; ++h; } dosmemput
39.    (v,X*Y,0xA0000    ); } else{       r.x.ax=
40.   0x13;             __dpmi_int(    0x10,&r); } }
41.               #elif defined(SDL)
42.           #include "SDL/SDL.h"
43.        SDL_Surface    *s; void
44.        F(int i){ if    (i){ SDL_SetColors(
45.     s,h,0,256);        SDL_UpdateRect
46.     (s,0,0,0,        0); } else {  SDL_Init(
47.       SDL_INIT_VIDEO); s=SDL_SetVideoMode
48.       (X,Y,8,0);        v=s->pixels; } }
```

```
49.                    #else
50.                 #include "curses.h"
51.               void F(i){ if(i){ for(y=0;
52.          y<X*Y                    ; y++)
53.           { move   (y/X,y%X);          addch
54.          ((*(v   +y)/      32)        [" ."
55.           ",:+"    "=@#"    ]); } ;   refresh
56.          (); }       else{           initscr
57.          (),x=      COLS&~1,X=x<X?x:X,y=
58.           LINES        &~1,Y=y<Y?y:Y; } }
59.            #endif
60.
61.  main(void)
62.  {
63.      F(0);
64.
65.      for (x=-X/2,y=-Y/2;y<Y/2;++x>=X/2?x=-X/2,y++:4)
66.                 {*q++ = sqrt(x*x+y*y);
67.
68.      *q++ = atan2(x,y);
69.
70.      }for (;n<j*2;l[n++]=0);
71.         for(;;)
72.         {
73.              a=l;z=l+j;e=l+j*2;
74.              if ((p+=B)>1){p=2-p;RI(l+j)}
75.                      else if (p<0){p=-p;RI(l)}
76.
77.              while(a<l+j) D=p**a+++(1-p)**z++,*e++=D;
78.              h=l+j*2;
79.
80.              for (J=P,z=v; z<v+X*Y;){
81.                   D = *J++;
82.                   *z++=fabs(sin((*J+++C[1])*1.5+D*C[0]+C[2]*sin(C[3]+D/C[4]))*255);
83.              }F(8);
84.
85.              C[2]+=B; f+=T/360; C[3]+=f;
86.
87.              if (f>T)
88.                   {C[1] += (f-T)/8;
89.
90.              if (f>T*2)
91.                   C[0]=sin(f)+sin(f*2)/2;
92.         }
93.      }
94.  }
```