```c
/****************************************************************************
 * adder.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Adds two numbers.
 *
 * Demonstrates use of CS50's library.
 ***************************************************************************/

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

int
main(void)
{
    // ask user for input
    printf("Give me an integer: ");
    int x = GetInt();
    printf("Give me another integer: ");
    int y = GetInt();

    // do the math
    printf("The sum of %d and %d is %d!\n", x, y, x + y);
}
```

```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
/****************************************************************************
 * ascii1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Displays the mapping between alphabetical ASCII characters and
 * their decimal equivalents using one column.
 *
 * Demonstrates casting from int to char.
 ***************************************************************************/

#include <stdio.h>


int
main(void)
{
    // display mapping for uppercase letters
    for (int i = 65; i < 65 + 26; i++)
        printf("%c: %d\n", (char) i, i);

    // separate uppercase from lowercase
    printf("\n");

    // display mapping for lowercase letters
    for (int i = 97; i < 97 + 26; i++)
        printf("%c: %d\n", (char) i, i);
}
```

```c
/****************************************************************************
 * ascii2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Displays the mapping between alphabetical ASCII characters and
 * their decimal equivalents using two columns.
 *
 * Demonstrates specification of width in format string.
 ***************************************************************************/

#include <stdio.h>


int
main(void)
{
    // display mapping for uppercase letters
    for (int i = 65; i < 65 + 26; i++)
        printf("%c  %d    %3d  %c\n", (char) i, i, i + 32, (char) (i + 32));
}
```

```c
/****************************************************************************
 * ascii3.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Displays the mapping between alphabetical ASCII characters and
 * their decimal equivalents.
 *
 * Demonstrates iteration with a char.
 ***************************************************************************/

#include <stdio.h>


int
main(void)
{
    // display mapping for uppercase letters
    for (char c = 'A'; c <= 'Z'; c = (char) ((int) c + 1))
        printf("%c: %d\n", c, (int) c);
}
```

```c
/****************************************************************************
 * battleship.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Prints a Battleship board.
 *
 * Demonstrates nested loop.
 ***************************************************************************/

#include <stdio.h>


int
main(void)
{
    // print top row of numbers
    printf("\n   ");
    for (int i = 1; i <= 10; i++)
        printf("%d  ", i);
    printf("\n");

    // print rows of holes, with letters in leftmost column
    for (int i = 0; i < 10; i++)
    {
        printf("%c  ", 'A' + i);
        for (int j = 1; j <= 10; j++)
            printf("o  ");
        printf("\n");
    }
    printf("\n");
}
```

```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
/****************************************************************************
 * buggy1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Should print 10 asterisks but doesn't!
 * Can you find the bug?
 ***************************************************************************/

#include <stdio.h>

int
main(void)
{
    for (int i = 0; i <= 10; i++)
        printf("*");
}
```

```c
/****************************************************************************
 * buggy2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Should print 10 asterisks, one per line, but doesn't!
 * Can you find the bug?
 ***************************************************************************/

#include <stdio.h>

int
main(void)
{
    for (int i = 0; i <= 10; i++)
        printf("*");
        printf("\n");
}
```

```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]);
}
```

```c
/****************************************************************************
 * capitalize.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Capitalizes a given string.
 *
 * Demonstrates casting and iteration over strings as arrays of chars.
 ***************************************************************************/

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


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

    // capitalize text
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        if (s[i] >= 'a' && s[i] <= 'z')
            printf("%c", s[i] - ('a' - 'A'));
        else
            printf("%c", s[i]);
    }
    printf("\n");
}
```

```c
/****************************************************************************
 * global.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Increments variables.
 *
 * Demonstrates use of global variable and issue of scope.
 ***************************************************************************/

#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)
{
    x++;
}
```

```c
/* http://www.ioccc.org/years.html */

#include "stdio.h"
#define e 3
#define g (e/e)
#define h ((g+e)/2)
#define f (e-g-h)
#define j (e*e-g)
#define k (j-h)
#define l(x) tab2[x]/h
#define m(n,a) ((n&(a))==(a))

long tab1[]={ 989L,5L,26L,0L,88319L,123L,0L,9367L };
int tab2[]={ 4,6,10,14,22,26,34,38,46,58,62,74,82,86 };

main(m1,s) char *s; {
    int a,b,c,d,o[k],n=(int)s;
    if(m1==1){ char b[2*j+f-g]; main(l(h+e)+h+e,b); printf(b); }
    else switch(m1-=h){
    case f:
        a=(b=(c=(d=g)<<g)<<g)<<g;
        return(m(n,a|c)|m(n,b)|m(n,a|d)|m(n,c|d));
    case h:
        for(a=f;a<j;++a)if(tab1[a]&&!(tab1[a]%((long)l(n))))return(a);
    case g:
        if(n<h)return(g);
        if(n<j){n-=g;c='D';o[f]=h;o[g]=f;}
        else{c='\r'-'\b';n-=j-g;o[f]=o[g]=g;}
        if((b=n)>=e)for(b=g<<g;b<n;++b)o[b]=o[b-h]+o[b-g]+c;
        return(o[b-g]%n+k-h);
    default:
        if(m1-=e) main(m1-g+e+h,s+g); else *(s+g)=f;
        for(*s=a=f;a<e;) *s=(*s<<e)|main(h+a++,(char *)m1);
    }
}
```

```
/*

  iUnlock v42.PROPER -- Copyright 2007 The dev team

  Credits: Daeken, Darkmen, guest184, gray, iZsh, pytey, roxfan, Sam, uns, Zappaz, Zf

  All code, information or data [from now on "data"] available
  from the "iPhone dev team" [1] or any other project linked from
  this or other pages is owned by the creator who created the data.
  The copyright, license right, distribution right and any other
  rights lies with the creator.

  It is prohibitied to use the data without the written agreement
  of the creator. This included using ideas in other projects
  (commercial or not commercial).

  Where data was created by more than 1 creator a written agreement
  from each of the creators has to be obtained.

  Punishment: Monkeys coming out of your ass Bruce Almighty style.

  [1] http://iphone.fiveforty.net/wiki/index.php?title=Main_Page
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <termios.h>
#include <errno.h>
#include <time.h>
#include "IOKit/IOKitLib.h"

#include "packets.h"

#define ever ;;
#define LOG stdout
#define SPEED 750000

#pragma pack(1)

#define BUFSIZE (65536+100)
unsigned char readbuf[BUFSIZE];

struct termios term;

//#define DEBUG_ENABLED 1

```

```c
#ifndef DEBUG_ENABLED
#define DEBUGLOG(x)
#else
#define DEBUGLOG(x) x
#endif

#define UINT(x) *((unsigned int *) (x))

const char * RE = "Why the hell are you reversing this app?! We said we were "\
"going to release the sources...";

void HexDumpLine(unsigned char *buf, int remainder, int offset)
{
  int i = 0;
  char c = 0;

  // Print the hex part
  fprintf(LOG, "%08x | ", offset);
  for (i = 0; i < 16; ++i) {
    if (i < remainder)
      fprintf(LOG, "%02x%s", buf[i], (i == 7) ? "  " : " ");
    else
      fprintf(LOG, "  %s", (i == 7) ? "  " : " ");
  }
  // Print the ascii part
  fprintf(LOG, " | ");
  for (i = 0; i < 16 && i < remainder; ++i) {
    c = buf[i];
    if (c >= 0x20 && c <= 0x7e)
      fprintf(LOG, "%c%s", c, (i == 7) ? " " : "");
    else
      fprintf(LOG, ".%s", (i == 7) ? " " : "");
  }

  fprintf(LOG, "\n");
}

void HexDump(unsigned char *buf, int size)
{
  int i = 0;

  for (i = 0; i < size; i += 16)
    HexDumpLine(buf + i, size - i, i);
  fprintf(LOG, "%08x\n", size);
}

int Checksum(CmdHeader * packet)
{
  int sum = 0x00030000;
  sum += packet->opcode;
  sum += packet->param_len;

  int len = packet->param_len;
  unsigned char * buf = ((unsigned char *)packet) + sizeof (CmdHeader);
  int i = 0;

  for (i = 0; i < len; ++i)
    sum += buf[i];
  return sum;
}

void SendCmd(int fd, void *buf, size_t size)
{
  DEBUGLOG(fprintf(LOG, "Sending:\n"));
  DEBUGLOG(HexDump((unsigned char*)buf, size));

  if(write(fd, buf, size) == -1) {
    fprintf(stderr, "Shit. %s\n", strerror(errno));
    exit(1);
  }
}

#define sendBytes(fd, args...) {\
  unsigned char sendbuf[] = {args}; \
  SendCmd(fd, sendbuf, sizeof(sendbuf)); \
}

int ReadResp(int fd)
{
  int len = 0;
  struct timeval timeout;
  int nfds = fd + 1;
  fd_set readfds;

  FD_ZERO(&readfds);
  FD_SET(fd, &readfds);

  // Wait a second
  timeout.tv_sec = 0;
  timeout.tv_usec = 500000;

  while (select(nfds, &readfds, NULL, NULL, &timeout) > 0)
    len += read(fd, readbuf + len, BUFSIZE - len);

  if (len > 0) {
    DEBUGLOG(fprintf(LOG, "Read:\n"));
    DEBUGLOG(HexDump(readbuf, len));
  }
  return len;
}

int InitConn(int speed)
{
  int fd = open("/dev/tty.baseband", O_RDWR | 0x20000 | O_NOCTTY);
  unsigned int blahnull = 0;
  unsigned int handshake = TIOCM_DTR | TIOCM_RTS | TIOCM_CTS | TIOCM_DSR;

  if(fd == -1) {
    fprintf(stderr, "%i(%s)\n", errno, strerror(errno));
    exit(1);
  }

  ioctl(fd, 0x2000740D);
  fcntl(fd, 4, 0);
  tcgetattr(fd, &term);

  ioctl(fd, 0x8004540A, &blahnull);
  cfsetspeed(&term, speed);
  cfmakeraw(&term);
  term.c_cc[VMIN] = 0;
  term.c_cc[VTIME] = 5;

  term.c_iflag = (term.c_iflag & 0xFFFFF0CD) | 5;
  term.c_oflag =  term.c_oflag & 0xFFFFFFFE;
  term.c_cflag = (term.c_cflag & 0xFFFC6CFF) | 0x3CB00;
  term.c_lflag =  term.c_lflag & 0xFFFFFA77;

  term.c_cflag = (term.c_cflag & ~CSIZE) | CS8;
  term.c_cflag &= ~PARENB;
  term.c_lflag &= ~ECHO;

  tcsetattr(fd, TCSANOW, &term);

  ioctl(fd, TIOCSDTR);
  ioctl(fd, TIOCCDTR);
  ioctl(fd, TIOCMSET, &handshake);

  return fd;
}

void RestartBaseband()
{
  kern_return_t   result;
  mach_port_t     masterPort;
```

```c

  result = IOMasterPort(MACH_PORT_NULL, &masterPort);
  if (result) {
    DEBUGLOG(printf("IOMasterPort failed\n"));
    return;
  }

  CFMutableDictionaryRef matchingDict = IOServiceMatching("AppleBaseband");
  io_service_t service = IOServiceGetMatchingService(kIOMasterPortDefault, matchingDict);
  if (!service) {
    DEBUGLOG(printf("IOServiceGetMatchingService failed\n"));
    return;
  }

  io_connect_t conn;
  result = IOServiceOpen(service, mach_task_self(), 0, &conn);
  if (result) {
    DEBUGLOG(printf("IOServiceOpen failed\n"));
    return;
  }

  result = IOConnectCallScalarMethod(conn, 0, 0, 0, 0, 0);
  if (result == 0)
    DEBUGLOG(printf("Baseband reset.\n"));
  else
    DEBUGLOG(printf("Baseband reset failed\n"));
  IOServiceClose(conn);
}

void SendGetVersion(int fd)
{
  sendBytes(fd, 0x60, 0x0D);
}

void GetVersion(int fd)
{
  SendGetVersion(fd);

  for (ever) {
    if(ReadResp(fd) != 0) {
      if(readbuf[0] == 0x0b)
        break;
    }
    SendGetVersion(fd);
  }

  VersionAck *ver = (VersionAck *) readbuf;
  DEBUGLOG(printf("Boot mode: %02X\n", ver->bootmode));
  DEBUGLOG(printf("Major: %d, Minor: %d\n", ver->major, ver->minor));
  DEBUGLOG(printf("Version: %s\n", ver->version));
}

void CFIStage1_2(int fd)
{
  CFIStage1Req req;
  req.cmd.cls = 0x2;
  req.cmd.opcode = BBCFISTAGE1;
  req.cmd.param_len = 0;
  req.checksum = Checksum ((CmdHeader*)&req);
  DEBUGLOG(printf("Sending CFIStage1 Request\n"));
  SendCmd(fd, &req, sizeof (CFIStage1Req));
  DEBUGLOG(printf("Receiving CFIStage1 response\n"));
  if (!ReadResp(fd)) {
    DEBUGLOG(fprintf(stderr, "Failed to receive CFIStage1 response\n"));
    exit(1);
  }
  CFIStage1Ack * cfi1resp = (CFIStage1Ack *) readbuf;
  cfi1resp->cmd.opcode = BBCFISTAGE2;
  cfi1resp->checksum = Checksum((CmdHeader*)cfi1resp);
  SendCmd(fd, cfi1resp, sizeof(CFIStage1Ack));
  if (!ReadResp(fd)) {
    DEBUGLOG(fprintf(stderr, "Failed to receive CFIStage2 response\n"));
    exit(1);
  }
}

void ReadSecpack(const char * FilePath, void * Buffer)
{
  FILE * fp = fopen(FilePath, "rb");
  if (fp == NULL) {
    perror(FilePath);
    exit(1);
  }

  fseek(fp, 0x1a4L, SEEK_SET);
  if (fread(Buffer, 1, 0x800, fp) != 0x800) {
    fprintf(stderr, "Error while reading the secpack content\n");
    free(Buffer);
    exit(1);
  }
  fclose(fp);
}

void SendBeginSecpack(int fd, void * Secpack)
{
  printf("Sending Begin Secpack command\n");

  BeginSecpackReq req;
  req.cmd.cls = 0x2;
  req.cmd.opcode = BBBEGINSECPACK;
  req.cmd.param_len = 0x800;
  memcpy(&req.data, Secpack, 0x800);
  req.checksum = Checksum((CmdHeader*)&req);
  SendCmd(fd, &req, sizeof (BeginSecpackReq));
  // Wait for the answer
  DEBUGLOG(printf("Reading answer\n"));
  while (!ReadResp(fd)) ;
}

void SendEndSecpack(int fd)
{
  printf("Sending End Secpack command\n");

  EndSecpackReq req;
  req.cmd.cls = 0x2;
  req.cmd.opcode = BBENDSECPACK;
  req.cmd.param_len = 0x2;
  req.unknown = 0;
  req.checksum = Checksum((CmdHeader*)&req);
  SendCmd(fd, &req, sizeof (EndSecpackReq));
  DEBUGLOG(printf("Reading answer\n"));
  ReadResp(fd);
}

void SendErase(int fd, int BeginAddr, int EndAddr)
{
  printf("Sending Erase command\n");

  EraseReq req;
  req.cmd.cls = 0x2;
  req.cmd.opcode = BBERASE;
  req.cmd.param_len = 8;
  req.low_addr = BeginAddr;
  req.high_addr = EndAddr;
  req.checksum = Checksum((CmdHeader*)&req);
  SendCmd(fd, &req, sizeof (EraseReq));
  sleep(1); // Give it some time
  DEBUGLOG(printf("Reading answer\n"));
  if (!ReadResp(fd)) {
    fprintf(stderr, "Ooops, something was wrong while erasing\n");
    exit(1);
  }

  printf("Waiting For Erase Completion...\n");
```

```c

  EraseAck * eraseack = (EraseAck *) readbuf;
  EraseStatusReq statusreq;
  statusreq.cmd.cls = 0x2;
  statusreq.cmd.opcode = BBERASESTATUS;
  statusreq.cmd.param_len = 2;
  statusreq.unknown1 = eraseack->unknown1;
  statusreq.checksum = Checksum((CmdHeader*)&statusreq);

  EraseStatusAck * erasestatusack = (EraseStatusAck *) readbuf;
  do {
    SendCmd(fd, &statusreq, sizeof(EraseStatusReq));
    DEBUGLOG(printf("Reading answer\n"));
    ReadResp(fd);
    erasestatusack = (EraseStatusAck *) readbuf;
  } while (erasestatusack->done != 1);

}

int ReadAddr(int fd, unsigned short int size)
{
  ReadReq req;

  req.cmd.cls = 0x2;
  req.cmd.opcode = BBREAD;
  req.cmd.param_len = 0x2;
  req.size = size;
  req.checksum = Checksum((CmdHeader*)&req);

  DEBUGLOG(printf("\nSending read request:\n"));
  SendCmd(fd, &req, sizeof(ReadReq));

  DEBUGLOG(printf("Receiving read response\n"));
  return ReadResp(fd);
}

void Seek(int fd, unsigned int addr)
{
  DEBUGLOG(printf("Sending seek command for addr %p\n", addr));
  SeekReq req;
  req.cmd.cls = 0x2;
  req.cmd.opcode = BBSEEK;
  req.cmd.param_len = 0x4;
  req.addr = addr;
  req.checksum = Checksum((CmdHeader*)&req);
  SendCmd(fd, &req, sizeof (SeekReq));
  DEBUGLOG(printf("Reading answer\n"));
  ReadResp(fd);
```

```c
}

void DumpReadBufToFile(FILE * fp)
{
  ReadAck * packet = (ReadAck*)readbuf;
  int len = packet->cmd.param_len;
  unsigned char * buf = &packet->first_char;
  fwrite(buf, len, 1, fp);
}

void Dump(int fd, FILE * fp)
{
  unsigned int addr = 0xa0000000;
  unsigned int nor_size = 0x400000; // the NOR is 4M (32Mbit)
  unsigned int page_size = 0x800;
  int i = 0;

  Seek(fd, addr);
  for (i = 0; i < nor_size; i += page_size) {
    DEBUGLOG(printf("Addr: %p\n", addr + i));
    ReadAddr(fd, page_size);
    DumpReadBufToFile(fp);
  }
}

void * ReadBL(const char * FilePath, int * Size)
{
  FILE * fp = fopen(FilePath, "rb");
  if (fp == NULL) {
    perror(FilePath);
    exit(1);
  }

  fseek(fp, 0, SEEK_END);
  int size = ftell(fp);
  fseek(fp, 0, SEEK_SET);

  void * buffer = malloc(size);

  if (fread(buffer, 1, size, fp) != size) {
    fprintf(stderr, "Error while reading the BL content\n");
    free(buffer);
    exit(1);
  }
  fclose(fp);
  *Size = size;
  return buffer;
}
```

```c

void * ReadFW(const char * FilePath, int Size)
{
  FILE * fp = fopen(FilePath, "rb");
  if (fp == NULL) {
    perror(FilePath);
    exit(1);
  }

  void * buffer = malloc(Size);
  fseek(fp, 0x9a4L + 0x20000, SEEK_SET);

  if (fread(buffer, 1, Size, fp) != Size) {
    fprintf(stderr, "Error while reading the FW content\n");
    free(buffer);
    exit(1);
  }
  fclose(fp);
  return buffer;
}


void SendWriteOnePage(int fd, unsigned char * Buffer, int Size)
{
  int size_to_write = Size > 0x800 ? 0x800 : Size;

  // Header, buffer, checksum
  int req_size = sizeof (CmdHeader) + size_to_write + 4;
  WriteReq * req = malloc(req_size);

  req->cmd.cls = 0x2;
  req->cmd.opcode = BBWRITE;
  req->cmd.param_len = size_to_write;
  memset(&req->first_char, 0, size_to_write);
  memcpy(&req->first_char, Buffer, size_to_write);
  *(unsigned int *)(&req->first_char + size_to_write) = Checksum((CmdHeader*)req);

  SendCmd(fd, req, req_size);
  DEBUGLOG(printf("Reading answer\n"));
  if (!ReadResp(fd)) {
    free(req);
    fprintf(stderr, "Ooops, something was wrong while Writing\n");
    exit(1);
  }
  free(req);
}

void SendWrite(int fd, unsigned char * Buffer, int Size, int Debug)
```

```c
{
  if (Debug) printf("Sending Write command\n");
  int cur_size = Size;
  int step = Size / 20;
  int last_progress = 0;

  while (cur_size > 0) {
    int progress = (Size - cur_size) / step;
    if (Debug && progress >= last_progress) {
      printf("%.2d%%\n", progress * 5);
      last_progress = progress;
    }
    SendWriteOnePage(fd, Buffer, cur_size);
    cur_size -= 0x800;
    Buffer += 0x800;
  }
}

// In progress ;)
void PatchingFW(unsigned char * Buffer)
{
  printf("Patching FW\n");

  if (Buffer[213740] != 0x04
    || Buffer[213741] != 0x00
    || Buffer[213742] != 0xa0
    || Buffer[213743] != 0xe1)
  {
    printf("Error in patch\n");
    exit(1);
  }
  Buffer[213740] = 0x00;
  Buffer[213741] = 0x00;
  Buffer[213742] = 0xa0;
  Buffer[213743] = 0xe3;

  memset(Buffer + 0x410, 0, 3);
  memset(Buffer + 0x800, 0, 16 * 10);
  memset(Buffer + 0xBFC, 0, 16 * 8);
  memset(Buffer + 0xFFC, 0, 16 * 8);
}

void ValidateFW(int fd, unsigned char * Buffer)
{
  printf("Validating the write command\n");
  Seek(fd, 0xA0020000);
  ReadAddr(fd, 0x800);

  ReadAck * packet = (ReadAck*)readbuf;
  unsigned char * buf = &packet->first_char;

  if (memcmp(buf, Buffer, 0x800)) {
    printf("FW differences found\n");
  } else {
    printf("FW are equal!\n");
  }
}

//void usage(char *prog)
//{
//  fprintf(stderr, "Usage: %s <fls file> [bl]\n", prog);
//  exit(1);
//}

void usage(char *prog)
{
  fprintf(stderr, "Usage: %s <fls file> <NOR file>\n", prog);
  exit(1);
}


void credit(void)
{
  printf("iUnlock v42.PROPER -- Copyright 2007 The dev team\n\n\n"\
         "Credits: Daeken, Darkmen, guest184, gray, iZsh, pytey, roxfan, Sam, uns, Zappaz, Zf\n\n" \
         "* Leet Hax not for commercial uses\n"\
         "  Punishment: Monkeys coming out of your ass Bruce Almighty style.\n\n"
         );
}

int main(int argc, char **argv)
{
  const char * hehe = RE;
  int fd;

  credit();

  if (argc != 3)
    usage(argv[0]);

  void * secpack = malloc(0x800);
  ReadSecpack(argv[1], secpack);

  void * fw = NULL;
  int fwsize = 0;
  fw = ReadBL(argv[2], &fwsize);
```

```c

  RestartBaseband();
  fd = InitConn(115200);

  GetVersion(fd);
  CFIStage1_2(fd);
  SendBeginSecpack(fd, secpack);
  SendErase(fd, 0xA0020000, 0xA03bfffe);
  Seek(fd, 0xA0020000 - 0x400);
  unsigned char foo[0x400];
  memset(foo, 0, 0x400);
  SendWrite(fd, foo, 0x400, false);
  SendWrite(fd, fw, fwsize, true);
  SendEndSecpack(fd);
  ValidateFW(fd, fw);
  printf("Completed.\nEnjoy!\n");
  free(fw);

  return 0;
}
```

```c
/****************************************************************************
 * math1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Computes a total but does nothing with it.
 *
 * Demonstrates use of variables.
 ***************************************************************************/

#include <stdio.h>

int
main(void)
{
    int x = 1;
    int y = 2;
    int z = x + y;
}
```

```c
/****************************************************************************
 * math2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Computes and prints an integral total.
 *
 * Demonstrates use of a format string.
 ***************************************************************************/

#include <stdio.h>

int
main(void)
{
    int x = 1;
    int y = 2;
    int z = x + y;
    printf("%d", z);
}
```

```c
/****************************************************************************
 * math3.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Computes and prints a floating-point total.
 *
 * Demonstrates loss of precision.
 ***************************************************************************/

#include <stdio.h>

int
main(void)
{
    float answer = 17 / 13;
    printf("%.2f\n", answer);
}
```

```c
/****************************************************************************
 * math4.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Computes and prints a floating-point total.
 *
 * Demonstrates use of floating-point math.
 ***************************************************************************/

#include <stdio.h>

int
main(void)
{
    float answer = 17 / 13.0;
    printf("%.2f\n", answer);
}
```

```c
/****************************************************************************
 * math5.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Computes and prints a floating-point total.
 *
 * Demonstrates use of casting.
 ***************************************************************************/

#include <stdio.h>

int
main(void)
{
    float answer = 17 / (float) 13;
    printf("%.2f\n", answer);
}
```

```c
/****************************************************************************
 * nonswitch.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Assesses the size of user's input.
 *
 * Demonstrates use of Boolean ANDing.
 ***************************************************************************/

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

int
main(void)
{
    // ask user for an integer
    printf("Give me an integer between 1 and 10: ");
    int n = GetInt();

    // judge user's input
    if (n >= 1 && n <= 3)
        printf("You picked a small number.\n");
    else if (n >= 4 && n <= 6)
        printf("You picked a medium number.\n");
    else if (n >= 7 && n <= 10)
        printf("You picked a big number.\n");
    else
        printf("You picked an invalid number.\n");
}
```

```
/****************************************************************************
 * positive1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Demands that user provide a positive number.
 *
 * Demonstrates use of do-while.
 ***************************************************************************/

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

int
main(void)
{
    // loop until user provides a positive integer
    int n;
    do
    {
        printf("I demand that you give me a positive integer: ");
        n = GetInt();
    }
    while (n < 1);
    printf("Thanks for the %d!\n", n);
}
```

```c
/****************************************************************************
 * positive2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Demands that user provide a positive number.
 *
 * Demonstrates use of bool.
 ***************************************************************************/

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

int
main(void)
{
    // loop until user provides a positive integer
    bool thankful = false;
    do
    {
        printf("I demand that you give me a positive integer: ");
        if (GetInt() > 0)
            thankful = true;
    }
    while (thankful == false);
    printf("Thanks for the positive integer!\n");
}
```

```c
/****************************************************************************
 * positive3.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Demands that user provide a positive number.
 *
 * Demonstrates use of !.
 ***************************************************************************/

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

int
main(void)
{
    // loop until user provides a positive integer
    bool thankful = false;
    do
    {
        printf("I demand that you give me a positive integer: ");
        if (GetInt() > 0)
            thankful = true;
    }
    while (!thankful);
    printf("Thanks for the positive integer!\n");
}
```

```
/****************************************************************************
 * progress1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Simulates a progress bar.
 *
 * Demonstrates sleep.
 ***************************************************************************/

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

int
main(void)
{
    // simulate progress from 0% to 100%
    for (int i = 0; i <= 100; i++)
    {
        printf("Percent complete: %d%%\n", i);
        sleep(1);
    }
    printf("\n");
}
```

```c
/****************************************************************************
 * progress2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Simulates a better progress bar.
 *
 * Demonstrates \r, fflush, and sleep.
 ***************************************************************************/

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

int
main(void)
{
    // simulate progress from 0% to 100%
    for (int i = 0; i <= 100; i++)
    {
        printf("\rPercent complete: %d%%", i);
        fflush(stdout);
        sleep(1);
    }
    printf("\n");
}
```

```c
/****************************************************************************
 * progress3.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Simulates a better progress bar.
 *
 * Demonstrates a while loop.
 ***************************************************************************/

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

int
main(void)
{
    int i = 0;

    /* simulate progress from 0% to 100% */
    while (i <= 100)
    {
        printf("\rPercent complete: %d%%", i);
        fflush(stdout);
        sleep(1);
        i++;
    }
    printf("\n");
}
```

```c
/****************************************************************************
 * return1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Increments a variable.
 *
 * Demonstrates use of parameter and return value.
 ***************************************************************************/

#include <stdio.h>


// function prototype
int increment(int a);


int
main(void)
{
    int x = 2;
    printf("x is now %d\n", x);
    printf("Incrementing...\n");
    x = increment(x);
    printf("Incremented!\n");
    printf("x is now %d\n", x);
}


/*
 * Returns argument plus one.
 */

int
increment(int a)
{
    return a + 1;
}
```

```c
/****************************************************************************
 * return2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Cubes a variable.
 *
 * Demonstrates use of parameter and return value.
 ***************************************************************************/

#include <stdio.h>


// function prototype
int cube(int a);


int
main(void)
{
    int x = 2;
    printf("x is now %d\n", x);
    printf("Cubing...\n");
    x = cube(x);
    printf("Cubed!\n");
    printf("x is now %d\n", x);
}


/*
 * Cubes argument.
 */

int
cube(int a)
{
    return a * a * a;
}
```

```c
/****************************************************************************
 * sizeof.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Reports the sizes of C's data types.
 *
 * Demonstrates use of sizeof.
 ***************************************************************************/

#include <stdio.h>

int
main(void)
{
    // some sample variables
    char c;
    double d;
    float f;
    int i;

    // report the sizes of variables' types
    printf("char: %d\n", sizeof(c));
    printf("double: %d\n", sizeof(d));
    printf("float: %d\n", sizeof(f));
    printf("int: %d\n", sizeof(i));
}
```