```c
/****************************************************************************
 * argv1.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Prints command-line arguments, one per line.
 *
 * Demonstrates use of argv.
 ****************************************************************************/

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


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

```c
/****************************************************************************
 * argv2.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Prints command-line arguments, one character per line.
 *
 * Demonstrates argv as a two-dimensional array.
 ****************************************************************************/

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


int main(int argc, string 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");
    }
    return 0;
}
```

```c
/*****************************************************************************
 * array.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Computes a student's average across 2 quizzes.
 *
 * Demonstrates C's math library.
 *****************************************************************************/

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

// number of quizzes per term
#define QUIZZES 2

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

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

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

    return 0;
}
```

```c
/****************************************************************************
 * ascii1.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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);

    return 0;
}
```

```c
/****************************************************************************
 * ascii2.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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);
    return 0;
}
```

```c
/*****************************************************************************
 * buggy3.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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);

    return 0;
}

/**
 * Swap arguments' values.
 */
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

```c
/*****************************************************************************
 * buggy4.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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);
    return 0;
}

/**
 * Tries to increment x.
 */
void increment(void)
{
    x++;
}
```

```c
/*****************************************************************************
 * buggy5.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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);
    return 0;
}

/**
 * Increments x.
 */
void increment(void)
{
    int x = 10;
    x++;
}
```

```c
/****************************************************************************
 * capitalize.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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");

    return 0;
}
```

```c
/****************************************************************************
 * global.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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);
    return 0;
}

/**
 * Increments x.
 */
void increment(void)
{
    x++;
}
```

```c
/*

  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"));
```

```
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;
```

```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));
```

```
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");
```

```c

  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);

```

```c
  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
/****************************************************************************
 * positive1.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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);
    return 0;
}
```

```c
/*****************************************************************************
 * positive2.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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");
    return 0;
}
```

```c
/****************************************************************************
 * positive3.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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");
    return 0;
}
```

```c
/*****************************************************************************
 * return.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * 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);
    return 0;
}

/**
 * Cubes argument.
 */
int cube(int a)
{
    return a * a * a;
}
```

```c
/****************************************************************************
 * string1.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Prints a given string one character per line.
 *
 * Demonstrates strings as arrays of chars and use of strlen.
 ****************************************************************************/

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

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

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

    return 0;
}
```

```c
/****************************************************************************
 * string2.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Prints a given string one character per line.
 *
 * Demonstrates strings as arrays of chars with slight optimization.
 ****************************************************************************/

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

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

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

    return 0;
}
```

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

                        int
                    X=320       ,Y=200,
                  n=0,m,      x,y,    j=1024;
                double      T=44.0      /7,P[
               333333      ],C[5]        ={ 0,3,
               0,0,8}       ,p=1,          B=11.0
                /630,       f=0,r    =      3,g
                 =7,b         =13,*q=P,    D,*J;
                 unsigned                 char
                   U[66666],*v=U,*h,l[5555]
                        ,c=0,*e,*a,*z;

                     #include <math.h>
                  #define R1(t)   t=(int)(t\
                *123456789           )%j; t/=j;
              #define                  Rl(C,t)\
              n++[C]         =          t*n/12;
               #define      RI(C)      B=-B; R1\
               (r)R1(g     )R1(b      )for(n\
                =0; n<j; ){ Rl(C    ,r)Rl\
                    (C,g)Rl(C    ,b)++n; }



        #ifdef __DJGPP__
          #include <sys/movedata.h>
                #include <dpmi.h>
                  #include <pc.h>
   #define          Q(u,v)         u##portb(0x3##v
     #define          W        ; Q(out,C9),*h++/4)
          void       F(int i){  __dpmi_regs r
           ; if(i){ for(; i>=0; i-=8)while(
                     ~Q(in,DA)
                  )&8^i); for(m=0,z
               =h+j; h    <z; m     ++){ Q(
          out,C8),m     )W W W; ++h; } dosmemput
   (v,X*Y,0xA0000   ); } else{       r.x.ax=
 0x13;            __dpmi_int(    0x10,&r); } }
                   #elif defined(SDL)
               #include "SDL/SDL.h"
           SDL_Surface    *s; void
          F(int i){ if   (i){ SDL_SetColors(
     s,h,0,256);         SDL_UpdateRect
     (s,0,0,0,        0); } else {  SDL_Init(
       SDL_INIT_VIDEO); s=SDL_SetVideoMode
       (X,Y,8,0);       v=s->pixels; } }
```

```
                  #else
               #include "curses.h"
             void F(i){ if(i){ for(y=0;
          y<X*Y                    ; y++)
         { move  (y/X,y%X);        addch
        ((*(v   +y)/     32)      [" ."
        ",:+"   "=@#"   ]); } ;  refresh
        (); }     else{          initscr
        (),x=     COLS&~1,X=x<X?x:X,y=
         LINES      &~1,Y=y<Y?y:Y; } }
          #endif

main(void)
{
    F(0);

    for (x=-X/2,y=-Y/2;y<Y/2;++x>=X/2?x=-X/2,y++:4)
                {*q++ = sqrt(x*x+y*y);

    *q++ = atan2(x,y);

    }for (;n<j*2;l[n++]=0);
        for(;;)
        {
            a=l;z=l+j;e=l+j*2;
            if ((p+=B)>1){p=2-p;RI(l+j)}
                        else if (p<0){p=-p;RI(l)}

            while(a<l+j) D=p**a+++(1-p)**z++,*e++=D;
            h=l+j*2;

            for (J=P,z=v; z<v+X*Y;){
                D = *J++;
                *z++=fabs(sin((*J+++C[1])*1.5+D*C[0]+C[2]*sin(C[3]+D/C[4]))*255);
            }F(8);

            C[2]+=B; f+=T/360; C[3]+=f;

            if (f>T)
                {C[1] += (f-T)/8;

            if (f>T*2)
                C[0]=sin(f)+sin(f*2)/2;
        }
    }
}
```