```c
/****************************************************************************
 * binary.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Displays a number in binary.
 *
 * Demonstrates bitwise operators.
 ***************************************************************************/

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


int
main(void)
{
    // prompt user for number
    int n;
    do
    {
        printf("Non-negative integer please: ");
        n = GetInt();
    }
    while (n < 0);

    // print number in binary
    for (int i = sizeof(int) * 8 - 1; i >= 0; i--)
    {
        int mask = 1 << i;
        if (n & mask)
            printf("1");
        else
            printf("0");
    }
    printf("\n");
}
```

```c
/****************************************************************************
 * bmp.h
 *
 * Computer Science 50
 * Problem Set 5
 *
 * BMP-related data types based on Microsoft's own.
 ***************************************************************************/

#include <stdint.h>


/*
 * Common Data Types
 *
 * The data types in this section are essentially aliases for C/C++
 * primitive data types.
 *
 * Adapted from http://msdn.microsoft.com/en-us/library/cc230309(PROT.10).aspx.
 * See http://en.wikipedia.org/wiki/Stdint.h for more on stdint.h.
 */

typedef uint8_t  BYTE;
typedef uint32_t DWORD;
typedef int32_t  LONG;
typedef uint16_t WORD;


/*
 * BITMAPFILEHEADER
 *
 * The BITMAPFILEHEADER structure contains information about the type, size,
 * and layout of a file that contains a DIB [device-independent bitmap].
 *
 * Adapted from http://msdn.microsoft.com/en-us/library/dd183374(VS.85).aspx.
 */

typedef struct
{
    WORD   bfType;
    DWORD  bfSize;
    WORD   bfReserved1;
    WORD   bfReserved2;
    DWORD  bfOffBits;
} __attribute__((__packed__))
BITMAPFILEHEADER;


/*
 * BITMAPINFOHEADER
 *
 * The BITMAPINFOHEADER structure contains information about the
 * dimensions and color format of a DIB [device-independent bitmap].
 *
 * Adapted from http://msdn.microsoft.com/en-us/library/dd183376(VS.85).aspx.
 */

typedef struct
{
    DWORD  biSize;
    LONG   biWidth;
    LONG   biHeight;
    WORD   biPlanes;
    WORD   biBitCount;
    DWORD  biCompression;
    DWORD  biSizeImage;
    LONG   biXPelsPerMeter;
    LONG   biYPelsPerMeter;
    DWORD  biClrUsed;
    DWORD  biClrImportant;
} __attribute__((__packed__))
BITMAPINFOHEADER;


/*
 * RGBTRIPLE
 *
 * This structure describes a color consisting of relative intensities of
 * red, green, and blue.
 *
 * Adapted from http://msdn.microsoft.com/en-us/library/aa922590.aspx.
 */

typedef struct
{
    BYTE  rgbtBlue;
    BYTE  rgbtGreen;
    BYTE  rgbtRed;
} __attribute__((__packed__))
RGBTRIPLE;
```

```
/****************************************************************************
 * endian.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Reads bf.bfSize from a BMP.
 *
 * Demonstrates endianness.
 ***************************************************************************/

#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>


int
main(int argc, char *argv[])
{
    // ensure proper usage
    if (argc != 2)
        return 1;

    // open file
    FILE *fp = fopen(argv[1], "r");
    if (fp == NULL)
        return 1;

    // seek to BITMAPFILEHEADER’s bfSize
    fseek(fp, 2, SEEK_SET);

    // read in BITMAPFILEHEADER’s bfSize
    uint32_t bfSize;
    fread(&bfSize, sizeof(bfSize), 1, fp);

    // print bfSize
    printf("\nbfSize: %d\n\n", bfSize);

    // return to start of file
    rewind(fp);

    // read in BITMAPFILEHEADER’s raw bytes
    uint8_t *buffer = malloc(14);
    fread(buffer, 1, 14, fp);

    // print field via cast
    printf("bfSize: %d\n\n", *((uint32_t *) (buffer + 2)));

    // print individual bytes in decimal
    printf("bfSize:   %d   %d   %d   %d\n",
           buffer[2], buffer[3], buffer[4], buffer[5]);

    // print individual bytes in hexadecimal
    printf("bfSize: 0x%x 0x%x 0x%x 0x%x\n",
           buffer[2], buffer[3], buffer[4], buffer[5]);

    // print individual bytes in binary
    printf("bfSize: ");
    for (int i = 2; i < 6; i++)
    {
        for (int j = 7; j >= 0; j--)
        {
            int mask = 1 << j;
            if (buffer[i] & mask)
```

```
                printf("1");
            else
                printf("0");
        }
    }
    printf("\n\n");

    // that’s all folks
    return 0;
}
```

```
/****************************************************************************
 * memory.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Demonstrates memory-related errors.
 *
 * problem 1: heap block overrun
 * problem 2: memory leak -- x not freed
 *
 * Adapted from
 * http://valgrind.org/docs/manual/quick-start.html#quick-start.prepare.
 ***************************************************************************/

#include <stdlib.h>


void
f(void)
{
    int *x = malloc(10 * sizeof(int));
    x[10] = 0;
}


int
main(void)
{
    f();
    return 0;
}
```

```c
/****************************************************************************
 * pointers1.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Prints a string, one character per line.
 *
 * Demonstrates strings as arrays.
 ***************************************************************************/

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


int
main(void)
{
    // prompt user for string
    printf("String please: ");
    char *s = GetString();
    if (s == NULL)
        return 1;

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

    // free string
    free(s);

    return 0;
}
```

```c
/****************************************************************************
 * pointers2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Prints a string, one character per line.
 *
 * Demonstrates pointer arithmetic.
 ***************************************************************************/

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


int
main(void)
{
    // prompt user for string
    printf("String please: ");
    char *s = GetString();
    if (s == NULL)
        return 1;

    // print string, one character per line
    for (int i = 0, n = strlen(s); i < n; i++)
        printf("%c\n", *(s+i));

    // free string
    free(s);

    return 0;
}
```

```c
/****************************************************************************
 * swap2.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Swaps two variables' values.
 *
 * Demonstrates (clever use of) bitwise operators.
 ***************************************************************************/

#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)
{
    *a = *a ^ *b;
    *b = *a ^ *b;
    *a = *a ^ *b;
}
```

```c
/****************************************************************************
 * tolower.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Converts an uppercase character to lowercase.
 *
 * Demonstrates bitwise operators.
 ***************************************************************************/

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


int
main(void)
{
    // prompt user for an uppercase character
    char c;
    do
    {
        printf("Uppercase character please: ");
        c = GetChar();
    }
    while (c < 'A' || c > 'Z');

    // print number in lowercase
    printf("%c\n", c | 0x20);

    // that's all folks
    return 0;
}
```

```c
/****************************************************************************
 * toupper.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Converts a lowercase character to uppercase.
 *
 * Demonstrates bitwise operators.
 ***************************************************************************/

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


int
main(void)
{
    // prompt user for a lowercase character
    char c;
    do
    {
        printf("Lowercase character please: ");
        c = GetChar();
    }
    while (c < 'a' || c > 'z');

    // print number in lowercase
    printf("%c\n", c & 0xdf);

    // that's all folks
    return 0;
}
```

```c
/****************************************************************************
 * uint.c
 *
 * Computer Science 50
 * David J. Malan
 *
 * Prints a signed 8-bit integer and an unsigned 8-bit integer.
 *
 * Demonstrates signed, fixed-width types.
 ***************************************************************************/

#include <stdint.h>
#include <stdio.h>


int
main(void)
{
    // declare and print signed 8-bit integer
    int8_t i = 0xff;
    printf("%d\n", i);

    // declare and print signed 8-bit integer
    uint8_t u = 0xff;
    printf("%d\n", u);
}
```