```
 1: /**************************************************************************
 2:  * binary.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Displays a number in binary.
 8:  *
 9:  * Demonstrates bitwise operators.
10:  **************************************************************************/
11:
12: #include <cs50.h>
13: #include <stdio.h>
14:
15:
16: int
17: main(int argc, char *argv[])
18: {
19:     // prompt user for number
20:     int n;
21:     do
22:     {
23:         printf("Non-negative integer please: ");
24:         n = GetInt();
25:     }
26:     while (n < 0);
27:
28:     // print number in binary
29:     for (int i = sizeof(int) * 8 - 1; i >= 0; i--)
30:     {
31:         int mask = 1 << i;
32:         if (n & mask)
33:             printf("1");
34:         else
35:             printf("0");
36:     }
37:     printf("\n");
38:
39:     // that's all folks
40:     return 0;
41: }
42:
```



```
 1: /**************************************************************************
 2:  * endian.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Reads bf.bfSize from a BMP.
 8:  *
 9:  * Demonstrates endianness.
10:  **************************************************************************/
11:
12: #include <stdio.h>
13: #include <stdlib.h>
14:
15:
16: int
17: main(int argc, char *argv[])
18: {
19:     // ensure proper usage
20:     if (argc != 2)
21:         return 1;
22:
23:     // open file
24:     FILE *fp = fopen(argv[1], "r");
25:     if (fp == NULL)
26:         return 1;
27:
28:     // seek to BITMAPFILEHEADER's bfSize
29:     fseek(fp, 2, SEEK_SET);
30:
31:     // read in BITMAPFILEHEADER's bfSize
32:     unsigned long bfSize;
33:     fread(&bfSize, sizeof(bfSize), 1, fp);
34:
35:     // print bfSize
36:     printf("\nbfSize: %ld\n\n", bfSize);
37:
38:     // return to start of file
39:     rewind(fp);
40:
41:     // read in BITMAPFILEHEADER's raw bytes
42:     unsigned char *buffer = malloc(14);
43:     fread(buffer, 1, 14, fp);
44:
45:     // print field via cast
46:     printf("bfSize: %ld\n\n", *((unsigned long *) (buffer + 2)));
47:
48:     // print individual bytes in decimal
49:     printf("bfSize:   %d   %d   %d   %d\n",
50:            buffer[2], buffer[3], buffer[4], buffer[5]);
51:
52:     // print individual bytes in hexadecimal
53:     printf("bfSize: 0x%x 0x%x 0x%x 0x%x\n",
54:            buffer[2], buffer[3], buffer[4], buffer[5]);
55:
56:     // print individual bytes in binary
57:     printf("bfSize: ");
58:     for (int i = 2; i < 6; i++)
59:     {
60:         for (int j = 7; j >= 0; j--)
61:         {
62:             int mask = 1 << j;
63:             if (buffer[i] & mask)
64:                 printf("1");
```

```
65:             else
66:                 printf("0");
67:         }
68:     }
69:     printf("\n\n");
70:
71:     // that's all folks
72:     return 0;
73: }
74:
```



```
 1: /****************************************************************************
 2:  * memory.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Demonstrates memory-related errors.
 8:  *
 9:  * problem 1: heap block overrun
10:  * problem 2: memory leak -- x not freed
11:  *
12:  * Adapted from
13:  * http://valgrind.org/docs/manual/quick-start.html#quick-start.prepare.
14:  ****************************************************************************/
15:
16: #include <stdlib.h>
17:
18:
19: void f()
20: {
21:     int *x = malloc(10 * sizeof(int));
22:     x[10] = 0;
23: }
24:
25:
26: int
27: main(int argc, char *argv[])
28: {
29:     f();
30:     return 0;
31: }
32:
```

swap2.c 1/1
lectures/7/src/

```
 1: /**************************************************************************
 2:  * swap2.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Swaps two variables' values.
 8:  *
 9:  * Demonstrates (clever use of) bitwise operators.
10:  **************************************************************************/
11:
12: #include <stdio.h>
13:
14:
15: // function prototype
16: void swap(int *, int *);
17:
18:
19: int
20: main(int argc, char *argv[])
21: {
22:     int x = 1;
23:     int y = 2;
24:
25:     printf("x is %d\n", x);
26:     printf("y is %d\n", y);
27:     printf("Swapping...\n");
28:     swap(&x, &y);
29:     printf("Swapped!\n");
30:     printf("x is %d\n", x);
31:     printf("y is %d\n", y);
32: }
33:
34:
35: /*
36:  * void
37:  * swap(int *a, int *b)
38:  *
39:  * Swap arguments' values.
40:  */
41:
42: void
43: swap(int *a, int *b)
44: {
45:     *a = *a ^ *b;
46:     *b = *a ^ *b;
47:     *a = *a ^ *b;
48: }
```

tolower.c 1/1
lectures/7/src/

```
 1: /**************************************************************************
 2:  * tolower.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Converts an uppercase character to lowercase.
 8:  *
 9:  * Demonstrates bitwise operators.
10:  **************************************************************************/
11:
12: #include <cs50.h>
13: #include <ctype.h>
14: #include <stdio.h>
15:
16:
17: int
18: main(int argc, char *argv[])
19: {
20:     // prompt user for an uppercase character
21:     char c;
22:     do
23:     {
24:         printf("Uppercase character please: ");
25:         c = GetChar();
26:     }
27:     while (c < 'A' || c > 'Z');
28:
29:     // print number in lowercase
30:     printf("%c\n", c | 0x20);
31:
32:     // that's all folks
33:     return 0;
34: }
35:
```

```
 1: /**************************************************************************
 2:  * toupper.c
 3:  *
 4:  * Computer Science 50
 5:  * David J. Malan
 6:  *
 7:  * Converts a lowercase character to uppercase.
 8:  *
 9:  * Demonstrates bitwise operators.
10:  **************************************************************************/
11:
12: #include <cs50.h>
13: #include <ctype.h>
14: #include <stdio.h>
15:
16:
17: int
18: main(int argc, char *argv[])
19: {
20:     // prompt user for a lowercase character
21:     char c;
22:     do
23:     {
24:         printf("Lowercase character please: ");
25:         c = GetChar();
26:     }
27:     while (c < 'a' || c > 'z');
28:
29:     // print number in lowercase
30:     printf("%c\n", c & 0xdf);
31:
32:     // that's all folks
33:     return 0;
34: }
35:
```