University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

EECS 61C, Fall 2003

HW 1

Goals

This assignment will give you practice compiling and executing C programs; it checks your understanding of the material in P&H Chapter 1, and number representation.

Background reading

K&R: Chapters 1-4.

P&H: Chapter 1, Chapter 4 sections 1 & 2.

Submitting Your Solution

Submit your solution online by 9am on September 3rd. Do this by creating a directory named hw1 that contains files named base2.print.c, bitcount.c, and cod.txt. "cod.txt" is a text file that you create containing your answers to the questions in P&H. (Note that capitalization matters in file names; the submission program will not accept your submission if your file names differ at all from those specified.) From within that directory, type "submit hw1".

This is not a partnership assignment; hand in your own work.

Problem 1

P&H problems 1.1-1.44 (these are very short questions), 1.47, 1.51, and 1.54.

P&H problems 4.1-4.8.

Problem 2

The program ~cs61c/lib/buggy.base2.print.c (listed below) is intended to print the binary (base 2) representation of the unsigned value stored in the variable numToPrintInBase2. It has bugs. Fix them, creating a file named base2.print.c by changing no more than three lines in buggy.base2.print.c. Also fill in the identification information at the top of the file. Don't delete or add any lines.

```
/*
   Name:
   Lab section time:
*/

#include <stdio.h>

int main ( ) {
   unsigned int numToPrintInBase2 = 1431655765;  /* alternating
1's and 0's */
   unsigned int exp = 1;
   int k;  /* can't declare variable in a loop header */


   /* Compute the highest storable power of 2 (2 to the 31th). */
   for (k=0; k<31; k++) {
      exp = exp * 2;
   }
   /* For each power of 2 from the highest to the lowest,
      print 1 if it occurs in the number, 0 otherwise. */
   for (k=31; !(k=0); k--) {
      if (numToPrintInBase2 >= exp) {
         printf ("%d", '1');
         numToPrintInBase2 = numToPrintInBase2 - exp;
      } else {
         printf ("%d", '0');
      }
      exp = exp / 2;
   }
   printf ("\n");
   return 0;
}
```

You should take advantage of this opportunity to learn more about the gdb debugger if you're not already familiar with it.

Problem 3

Write a function named bitCount that returns the number of 1-bits in the binary representation of its unsigned integer argument. Add your function to the following program, which is available online in ~cs61c/lib/bitcount.c, fill in the identification information, and run the completed program.

```
/*
   Name:
   Lab section time:
*/

#include <stdio.h>

int bitCount (unsigned int n);

int main ( ) {
   printf ("# 1-bits in base 2 rep of %u = %d, should be 0\n",
      0, bitCount (0));
   printf ("# 1-bits in base 2 rep of %u = %d, should be 1\n",
      1, bitCount (1));
   printf ("# 1-bits in base 2 rep of %u = %d, should be 16\n",
      1431655765, bitCount (1431655765));
   printf ("# 1-bits in base 2 rep of %u = %d, should be 1\n",
      1073741824, bitCount (1073741824));
   printf ("# 1-bits in base 2 rep of %u = %d, should be 32\n",
      4294967295, bitCount (4294967295));
   return 0;
}

/* Your bit count function goes here. */
```