**Exercise 3-1.** Our binary search makes two tests inside the loop, when one would suffice (at the price of more tests outside.) Write a version with only one test inside the loop and measure the difference in run-time.

**Exercise 3-2.** Write a function escape(s,t) that converts characters like newline and tab into visible escape sequences like \n and \t as it copies the string t to s. Use a switch. Write a function for the other direction as well, converting escape sequences into the real characters.

**Exercise 3-3.** Write a function expand(s1,s2) that expands shorthand notations like a-z in the string s1 into the equivalent complete list abc...xyz in s2. Allow for letters of either case and digits, and be prepared to handle cases like a-b-c and a-z0-9 and -a-z. Arrange that a leading or trailing - is taken literally.

**Exercise 3-4.** In a two's complement number representation, our version of itoa does not handle the largest negative number, that is, the value of n equal to -(2wordsize-1). Explain why not. Modify it to print that value correctly, regardless of the machine on which it runs.

**Exercise 3-5.** Write the function itob(n,s,b) that converts the integer n into a base b character representation in the string s. In particular, itob(n,s,16) formats s as a hexadecimal integer in s.

**Exercise 3-6.** Write a version of itoa that accepts three arguments instead of two. The third argument is a minimum field width; the converted number must be padded with blanks on the left if necessary to make it wide enough.