This project is a two part programming assignment. For each part read and follow the instructions carefully. You should turn in a printout of both of your programs and you should submit your source code as usual to the TA via email at `pliu2@umbc.edu`. All work is due on November 26, 2002 at 5:30 pm.

Part 1: Testing for Prime Numbers

Prime numbers have been studied by mathematicians for many years. A prime number is an integer that has no divisors other than 1 and itself (for example, the integers 2, 3, 5, 7, and 11 are prime). Write a modular program called `prime.c` that finds the smallest divisor of a number or determines that the number is prime.

To determine whether an integer n is prime, your program should test integers that are smaller than n until it finds an integer that is a divisor of n. The integer values to be tested should be limited to 10,000 or less.

The program should prompt the user for the integer n to be tested. Note that n must be greater than 1 to be valid and less than or equal to $N = 10,000$. Error checking should be included as well as repition to get valid input from the user.

Your program must contain two functions in addition to `main`; one for finding the smallest divisor and one to determine if a number is even. These functions should be named `findDivisor` and `isEven` respectively. `findDivisor` should take one integer as input and return one integer that is the smallest divisor of the input. The `isEven` function should take one integer as input and return 1 if the integer is even and 0 otherwise.

Note that if any number is even, then its smallest divisor other than 1 is 2. If a number is not even, than the odd number starting from 3 should be tested to see if they divide the number perfectly. Your program should take this information into account.

Your program should print the smallest divisor of a given number, along with that number, if the number is not prime. Otherwise, your program should print the given number and that it is prime.

Part 2: Factoring an Integer

Factoring a number is another interesting mathematical problem. Write a program called `factor.c` that takes an integer value and factors that number into its prime factors. The program should contain a new function called `factor` and should incorporate both functions from part 1, `findDivisor` and `isEven`. The `factor` function should take one integer as input, and return nothing (i.e., `void`).

The `findDivisor` function returns the smallest divisor of a number. In order to find all of the factors, `findDivisor` should be called iteratively on the result of the number divided by its smallest divisor. As an example, consider the number 84.

```
84 = 2 x 42
42 = 2 x 21
21 = 3 x 7
 7 = 1 x 7
```

Therefore, the factorization of 84 is

```
84 = 2 x 2 x 3 x 7
```

The program should ask the user for an integer to factor. The input should be checked for validity and re-entered if incorrect. You should ensure that the input number is greater than 1 and less than $N = 10,000$. The output of the program should be the input number and the number's factorization, as seen above.