Lab Assignment 1.

Due Weds May 26 at 11PM. Labs Mon May 17 and Mon May 24.

Data Structures Engr 4892, 2004

0 BigNum

In C++ integers are usually limited to holding values up to around 9 digits (int). For some applications (notably cryptography) much larger numbers (hundreds of digits) are needed with no loss of precision. One way to represent natural numbers is as a linked list of digits in some base. For this assignment we will use a base of 2^{32} (thus each "digit" can be between 0 and $2^{32}-1$, inclusive). We will write the digits in an order so that the closest digit to the head is the least significant, while the closest digit to the tail is the most significant. In order that each number have a unique representation, we will assume that the most significant digit will not be 0. A consequence of this is that the number 0 will be represented by an empty list. For simplicity we will ignore the existence of negative numbers. We will also assume that the unsigned type uses 32 bits and thus can represent exactly the set $\{0, 1, ..., 2^{32}-1\}$.

The number represented by a link p is

$$\text{p->digit} + 2^{32} \times \text{p->next->digit} + 2^{64} \times \text{p->next->next->digit} + \cdots$$

or, to be a bit more precise, it is $f(p)$ where f is defined by

$$f(p) = \begin{cases} 0 & \text{, if } p \text{ is null} \\ p\text{->digit} + 2^{32} \times f(p\text{->next}) & \text{, otherwise} \end{cases}$$

Write a subroutine, add1, that takes a single digit and a linked list representing a number and alters the list to be the sum of the original value of the list and the digit. The declaration of the subroutine will be

---

void **add1**(unsigned **digit**, BigNumNode* &**sum**, bool &**ok**) ;

Write a subroutine, add, that takes two linked lists representing two numbers and alters the second so that its final value represents the sum of the initial values of the two numbers. The first list should be unaltered. The declaration of the subroutine will be

---

void **add**(BigNumNode* **addend**, BigNumNode* &**sum**, bool &**ok**) ;

Write a subroutine, mult1, that takes a linked list representing a natural number and a positive digit (an integer from 1 to $2^{32}-1$ inclusive) and computes a new list that represents the product of the two numbers. The declaration of the subroutine will be

```
void mult1(unsigned digit, BigNumNode* multiplicand, BigNumNode* &product,
bool &ok) ;
```

Write a subroutine, mult, that takes two linked lists representing two numbers and computes a new list that represents the product of the two numbers. The declaration of this subroutine will be

```
void mult(BigNumNode* multiplicand, BigNumNode* multiplier, BigNumNode*
&product, bool &ok) ;
```

Notes:

- I will provide you with the BigNumNode type, defined by

```
struct BigNumNode {
      unsigned digit ;
      BigNumNode *next ; } ;
```

- The null (0) pointer will mark the last link of every list.
- Every digit will be an integer from $\{0, 1, ..., 2^{32} - 1\}$.
- Leading 0 digits should not be produced and need not be anticipated.
- It follows that the number 0 will be represented by a null pointer (an empty list)!
- As a precondition to mult1, the digit parameter will not be 0. It follows that the output list will be as longer as or longer than the input list.
- Don't forget that the least significant digit is at the *head* of the list.
- In each case, set ok to false if you run out of heap space, and to true if you do not.
- I will provide a subroutine to compute the 64 bit product (as two unsigned variables) of two 32 bit numbers, a subroutine to delete unneeded lists, and a subroutine to copy a list.

Put your code in a file bignum.cpp and submit it as assignment 1.