COP 3503 - Programming Assignment # 4

Integer Multiplication using the standard multiplication algorithm and Karatsuba's algorithm for fast multiplication of large integers

Assigned: Feb 25, 2014 (Tuesday)

Due: March 13, 2014 (Thursday) at 11:55 PM WebCourses time

Problem: Integer multiplication of two binary numbers

Given two binary integers, x and y, compute their product x*y. For example, if *x = 1001* (9 in base 10) and *y = 1100* (12 in base 10) , then *x*y = 1101100* (108 in base 10).

Algorithm 1 – Standard multiplication algorithm

This is the grade-school multiplication algorithm, also known as long multiplication. This algorithm is illustrated below for *1001 * 1100*.

```
          1 0 0 1     (9)
       *  1 1 0 0     (12)
        ----------
          0 0 0 0
        0 0 0 0
      1 0 0 1
+ 1 0 0 1
----------------
  1 1 0 1 1 0 0     (108)
```

Algorithm 2 – Karatsuba's divide and conquer algorithm

This algorithm is described in Section 5.4 of the textbook.

Implementation Notes

Represent the binary numbers using character arrays of '0' or '1'. The algorithms must work with the bitstrings directly – e.g., do not convert to base 10 and then multiply the numbers. However, for debugging purposes you may choose to implement a method to convert to base 10.

Karatsuba's algorithm uses addition and subtraction. Implement a binary addition method using the "grade-school" addition algorithm. You may also implement a similar binary subtraction method which "borrows" from the next non-zero position to the left. Alternatively, implement subtraction "two's complement arithmetic" as the following example shows.

Suppose we want to compute 17 – 13, i.e. 00010001 – 00001101. The answer is 4, i.e., 00000100.

1) Change 00001101 (13 in base 10) to a negative number in "two's complement representation" by inverting each bit and adding 1. So, 00001101 becomes 11110010 + 1 = 11110011.
2) Compute 17 + (-13), i.e., 00010001 + 11110011, using the "grade-school" addition algorithm. There will be a "carry" which spills over into the left-most (n + 1)th bit that we ignore.

```
Carry: 1 1 1       1 1
       0 0 0 1 0 0 0 1   (17)
     + 1 1 1 1 0 0 1 1   (-13)
     -----------------
       0 0 0 0 0 1 0 0   (4)
```

Input and Output

Input

Input is from **mult.txt**. The first line is the number of multiplications, *k*. The rest of the file contains the *k* multiplications. Each multiplication contains two lines corresponding to the two numbers to be multiplied. A line of a multiplication begins with the number of bits *n* of the number to be multiplied followed by a space and *n* bits (0s or 1s).

Output

Output is to **System.out**. Each multiplication *j* in the input (where multiplications are labeled *0* through *k-1*) produces two lines of output, each of which contain the result of multiplying the two numbers of multiplication *j* – the first line is the result of using the standard multiplication algorithm and the second line is the result of Karatsuba's algorithm.

Sample input

```
3
4 1001
4 1100
5 11111
5 11111
33 111010011000001011001111101001101
34 1000110001010011110100111001111011
```

Sample output

```
1101100
1101100
1111000001
1111000001
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
```

Restrictions on Source Code

- Submit a file named **Multiplication.java** which defines a *public class Multiplication* which defines a *public static void main(String[])* method. This method Reads the input file "mult.txt" and produces the desired output to System.out. You may submit other Java files as well.
- Do not put any of your your Java files in a package. Omit the package statement to leave them in the default package.
- Your program must compile using Java 7.0 or later. It's okay to develop your program using the IDE of your choice, although Eclipse is recommended. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date.

Deliverables

You must submit Multiplication.java and any additional Java files to WebCourses by 11:55 PM on Thursday, March 13, 2014. You must send your source files as an attachment using the "Add Attachments" button. Assignments that are typed into the submission box will not be accepted. Assignments that are 1 day late are deducted 25% of the points received. Assignments more than 1 day late are not accepted. Programs that do not compile will receive no credit.