csce350 — Data Structures and Algorithms
Fall 2020 — Project 3: A Return to Fifth Grade?

Assigned: November 3
Due: November 24, 11:55pm

For this assignment, you will use C++ to implement two different algorithms for multiplying very large integers. The purpose of the assignment is to give you some experience translating a non-trivial algorithm from pseudocode to a working implementation.

Your task

Write a program that does these steps, in this order:

1. Read two digit strings a and b, separated by a $\star$ character, from standard input. The numbers a and b may be very large, potentially thousands of digits each. Either or both of the numbers may be zero.
2. Multiply a and b using the $\Theta(n^2)$ time brute force algorithm described in class. The text book mentions this algorithm on page 187 as the "pencil and paper" algorithm. It's also the method that is usually taught to small children.
3. Output the text "`** Brute Force: `", followed by the answer from Step (2).
4. Multiply a and b again using the divide-and-conquer Karatsuba multiplication algorithm.
5. Output the text "`** Karatsuba: `", followed by the answer from Step (4).
6. Stop.

That's it. Just multiply the two numbers using both algorithms. Here are some examples:

Sample Input 1
```
1234*5678
```

Sample Output 1
```
** Brute Force: 7006652
** Karatsuba: 7006652
```

Sample Input 2
```
123456789123456789*987654321987654321
```

Sample Output 2
```
** Brute Force: 121932631356500531347203169112635269
** Karatsuba: 121932631356500531347203169112635269
```

Sample Input 3
```
4802064505264608796033165326358 46*21789169871971286
```

Sample Output 3
```
** Brute Force: 1046329992413743128861878710649155077951360631 7956
** Karatsuba: 1046329992413743128861878710649155077951360631 7956
```

Sample Input 4

```
32780765971218003881977971970736230585762785519046116 38*0
```

Sample Output 4

```
** Brute Force: 0
** Karatsuba: 0
```

For **this project only**, additional lines of output are allowed, as long as they do not begin with `**`. Thus, you can include additional outputs to show the progress of the algorithm. Such outputs are likely to helpful to you as you develop the program, and may be useful in grading, especially when determining partial credit.

Some additional comments:

- The most common mistake for this kind of program is the try to use an `int` variable to store the input, the output, or any of the values computed along the way. Remember that many C++ compilers use 32 bits of memory to store `int` variables. This means that for numbers larger than about $2^{32} = 4,294,967,296$, things will begin to fail (without any error message) because of overflow. Because you want a program that works for numbers much larger than 2^{32}, you *must* store numbers as *vectors of digits* at **every** step along the way.
- You may use (and indeed, probably should) use the `vector` class for the "array" that holds the digits in each number you manipulate. This simplifies some aspects of the problem because it is much easier to change the size of a vector than of an C++ array.
- You may generate additional debugging output for each algorithm if you like, as long as the two lines required in Steps (3) and (5) appear eventually. Such extra outputs may be useful in awarding partial credit if your solution is not fully correct, provided they can be deciphered.
- Watch out for the case that a and b have different numbers of digits. The easiest way to handle this is to "pad" the shorter number with leading 0's to equalize the lengths.

What to Submit: You should submit, using the department's dropbox website, a single C++ source file named containing all of the code for your program. I will compile this program using this command line:

```
g++ -Wall -std=c++11 yourfile.cpp
```