csce350 — Data Structures and Algorithms
Fall 2019 — Project 3

Assigned: October 30
Due: November 19, 11:54pm

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

Your Task You should write a C++ program that does these things:

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 some $\Theta(n^2)$ time brute force algorithm. The text book mentions one such algorithm on page 187 as the "pencil and paper" algorithm. Any method typically taught to small children will work.
3. Output the text "`B: `", 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 "`K: `", followed by the answer from Step (4).

That's it. Just multiply the two numbers using both algorithms.

Sample Input 1
```
1234*5678
```
Sample Output 1
```
B: 7006652
K: 7006652
```

Sample Input 2
```
123456789123456789*987654321987654321
```
Sample Output 2
```
B: 121932631356500531347203169112635269
K: 121932631356500531347203169112635269
```

Sample Input 3
```
480206450526460879603316532635846*21789169871971286
```
Sample Output 3
```
B: 10463299924137431288618787106491550779513606317956
K: 10463299924137431288618787106491550779513606317956
```

Hints A few possibly helpful comments:

- The most common error for this kind of program is to try to use an `int` variable to store the input, the output, or values computed along the way. Remember that most 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 because of overflow. **These kinds of overflow problems will not generate any error messages.** 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 STL `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 will want to think carefully about the **order** in which the digits are stored in your `vector` of digits. Which digit goes in position `0`? Which digit goes in position `i`? There are two choices for how to organize things, both of which can work just fine, but each has advantages and disadvantages.
- When you divide the two numbers a and b into a_0, a_1, b_0 and b_1, be careful to use the **same value of** m for both a and b.
- 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.
- 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.
- For calibration purposes, my solution has 266 lines of code, including comments.

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
```